



中国科学院大学  
University of Chinese Academy of Sciences

## 电子系统设计实验报告

题目：基于 FPGA 的便携式心电图机设计

指导教师： 贾颖新、孙忆南

组长： 潘钊滢

成员： 陈旧、马金戈、马千里、曾志鸿 （按姓氏排序）

专业： 电子信息工程

2020 年 5 月

# 目录

一、选题简介 .....	3
二、系统实现方案.....	3
三、硬件设计与实现.....	4
1. 供电模块设计.....	4
2. 稳压模块设计.....	5
3. 心电采集模块设计 .....	6
4. PCB 绘制、焊接与调试 .....	8
(1) PCB 绘制.....	8
(2) 焊接+初步调试电压.....	11
(3) PCB 调试.....	14
四、软件设计与实现.....	16
1. FPGA 处理 .....	16
(1) SPI 通信.....	16
(2) 数字滤波.....	21
(3) 蓝牙通信.....	27
2. 小程序 .....	30
(1) 页面布局.....	30
(2) 蓝牙通信.....	30
(3) 动态显示.....	36
(4) 小程序整合 .....	41
(5) 心率计算.....	41
(6) 小程序调试.....	42
五、人员分工 .....	44
六、作品展示操作说明 .....	44
七、经验总结 .....	47
八、待优化与待改进方案 .....	51
九、参考文献 .....	51

## 一、选题简介

本课程设计中，我们希望实现的是基于 FPGA 的便携式心电图机，系统的主要功能是采集、处理、分析和显示心电图信号，最终在微信小程序上显示心电图以及心率数据。

## 二、系统实现方案

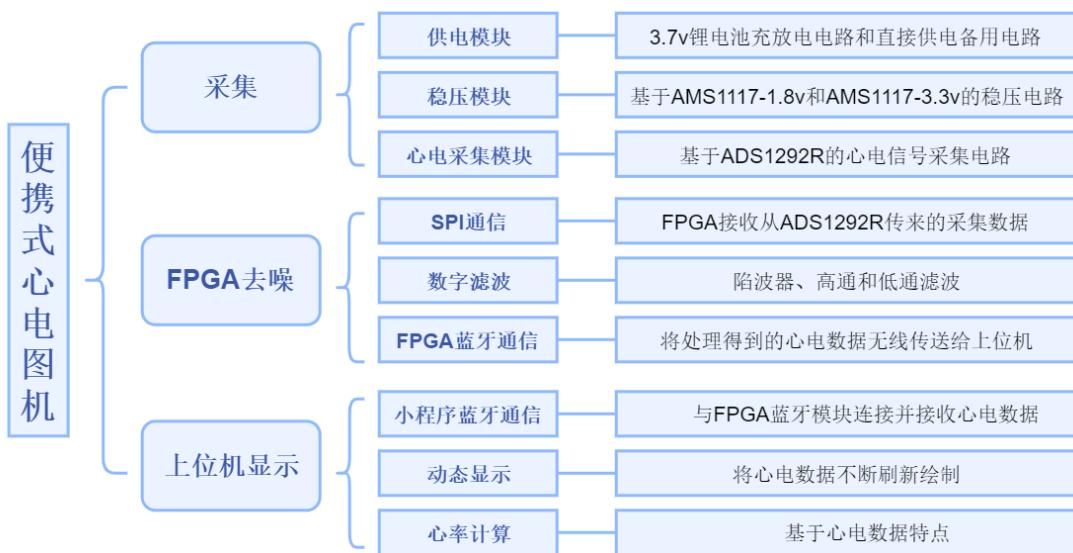


Figure 1. 系统设计框图

整体设计分为三个部分：采集、FPGA 去噪、上位机显示。

采集部分包括供电模块、稳压模块和心电采集模块。供电模块设计了 3.7V 锂电池充放电电路和 micro-usb 口直接供电备用电路；稳压模块是基于 AMS1117-1.8V 和 AMS1117-3.3V 的稳压电路；心电采集模块是基于 ADS1292R、用两个裸露的圆形焊盘做电极的心电信号采集电路。

FPGA 软件设计包括 SPI 通信、数字滤波和蓝牙通信。SPI 通信即 FPGA 板卡以 SPI 协议接收从 ADS1292R 传来的采集数据；数字滤波由高通、低通滤波和陷波器三块组成，分别去除基线漂移、肌电噪声和工频干扰，实际实验时采用一个带通滤波器实现高通滤波和低通滤波；蓝牙通信是将处理得到的心电数据通过蓝牙无线传送给上位机（即微信小程序）。

上位机软件设计包括基于微信小程序的蓝牙通信和动态显示。蓝牙通信是完成与 FPGA 蓝牙模块的连接并接收心电数据；动态显示是将心电数据通过小程序的 canvas 组件不断刷新绘制；此外还有根据心电数据特点进行心率计算等拓展功能。

### 三、硬件设计与实现

#### 1. 供电模块设计

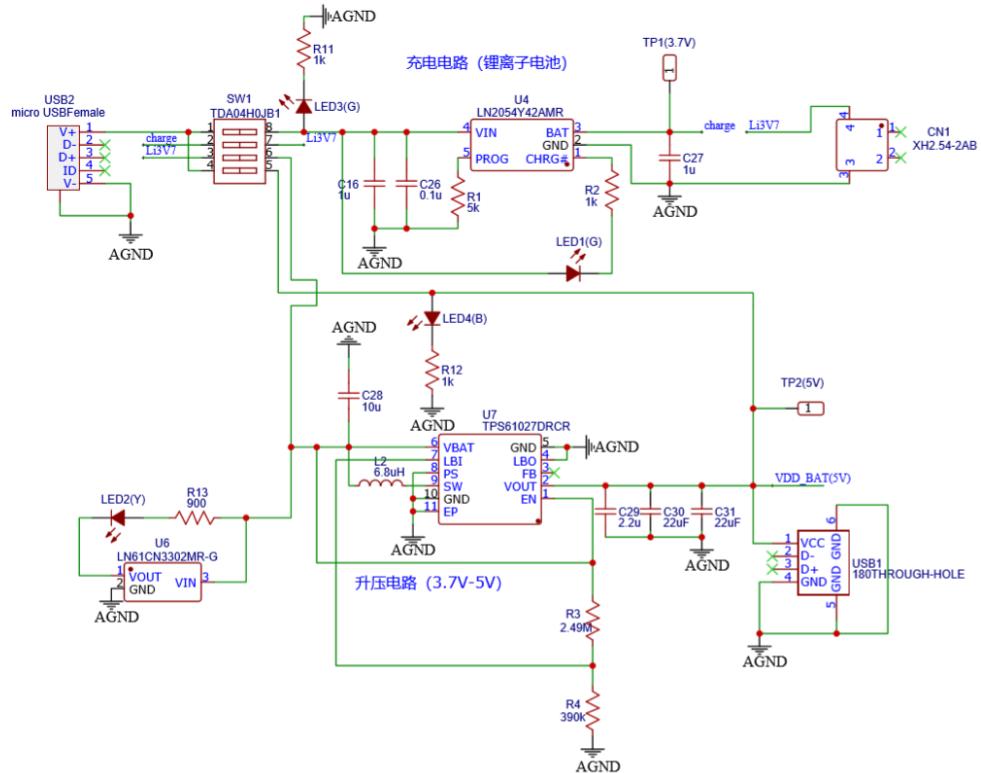


Figure 2. 供电电路原理图

通过 micro-USB 接口提供 5V 电压（充电宝或者手机充电器都可，输出电压为 5V）；注意只用作充电的话 D+、D- 和 ID 都悬空即可；多出来的第六个引脚是外壳，接地即可；

利用 LN2054Y42AMR 芯片[1]为 3.7V 锂离子电池充电；锂离子电池可以通过插针 CN1 连接到 PCB 板上；

利用 LN61CN3302MR-G 作为电压检测芯片[2]，用于检测锂离子电池是否电压不足，即是否需要充电。低于 3.3V 时 LED2 应由灭变亮；注意这部分的 R13 是一个限流电阻，稍大一点没关系，设计中没考虑到  $900\Omega$  很难买到，后改用  $1k\Omega$  电阻；

开始选择的升压芯片 TPS61099[3]太小，焊接很困难；后改用 TPS61027DRCR 作为升压芯片[4]（实际也很困难，为 QFN 封装），将锂电池的 3.7V 电压转为 5V；C30、C31 是用来替换原来的  $47\mu F$  的直插式电解电容，老师建议这个容量就没必要用电解电容，改为了 0805 封装的两个  $22\mu F$  贴片陶瓷电容作为替代；电感 L2 的选择需要特别考虑，要过电流，需要计算电感的最大电流，电感有最大电流指标，根据这个去商城选型确定封装；

R3、R4 是根据芯片手册计算得到的，本设计中  $V_{BAT}$  为 3.7V 锂离子电池供电电压，注意实际买不到  $2.5M\Omega$  电阻， $2.49M\Omega$  很容易买到，在电阻的参数确定时要灵活处理；

$$V_{BAT} = 3.7V$$

$$R_1 = R_2 \times \left( \frac{V_{BAT}}{V_{LBI-threshold}} - 1 \right) = 390k\Omega \times \left( \frac{V_{BAT}}{500mV} - 1 \right) = 2496k\Omega$$

输出的 5V 电压作为稳压电路的输入，同时也通过 USB 接口作为 FPGA 板卡的供电；

此外，设计了备用的直接供电电路（即 USB2，拨码开关第四位，LED4(B)，R12 组成的电路，直接输出到  $V_{VDD-BAT}(5V)$ ；这里注意指示灯不能直接串联进去，可以串联电阻接地后并联在电路中；

使用四位拨码开关来控制充电、放电和直接供电不同模式；这里拨码开关的封装没有注意，导致器件买不到，只好找可以短一点引脚且引脚间距合适的拨码开关来替代。

设计了 TP1 和 TP2 两个测试点用于检测电压是否符合要求；TP 点在 PCB 上用圆形焊盘或者通孔可以实现，原理图上可以不反映出来，只在 PCB 上体现即可；

## 2. 稳压模块设计

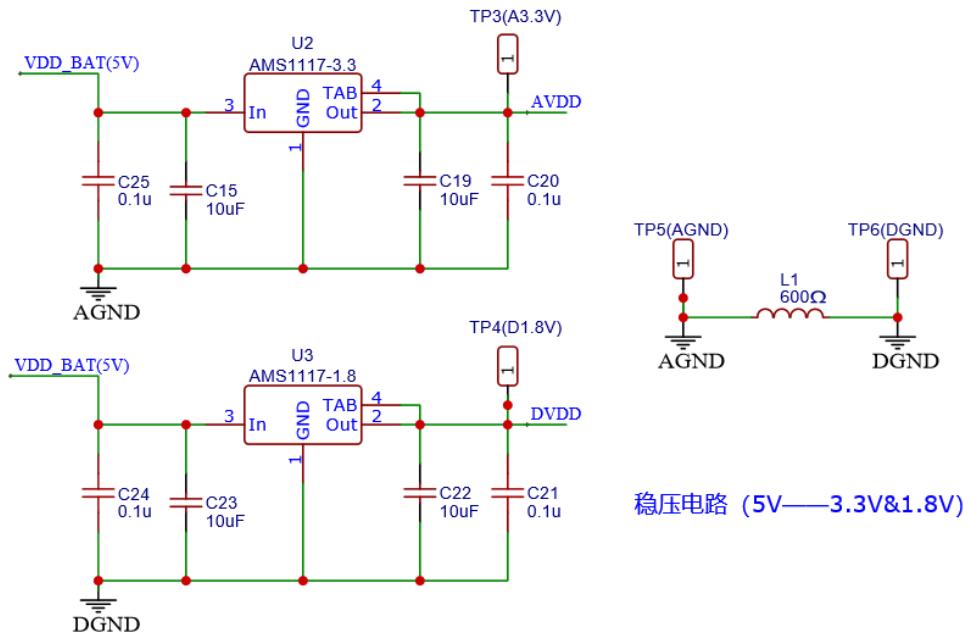


Figure 3. 稳压电路原理图

AMS1117 是稳压芯片[5]，将 5V 输入电压转化为需要的 3.3V 和 1.8V，分别为 ADS1292R 芯片提供模拟电压和数字电压；

这里需要注意的是查阅到的参考电路中 AMS1117 芯片只有三个引脚，但是原理图的封装都是 SOT-223（四个引脚），查阅芯片手册才知道 2 脚和 4 脚是连在一起的；

ADS1292R 芯片[6]有模拟电源引脚和数字引脚，为了减少干扰，模拟和数字电源要分开供电，同时模拟地和数字地也要分开。于是本设计中采用磁珠（对应原理图中的 L1）对模拟地和数字地进行隔离。

### 3. 心电采集模块设计

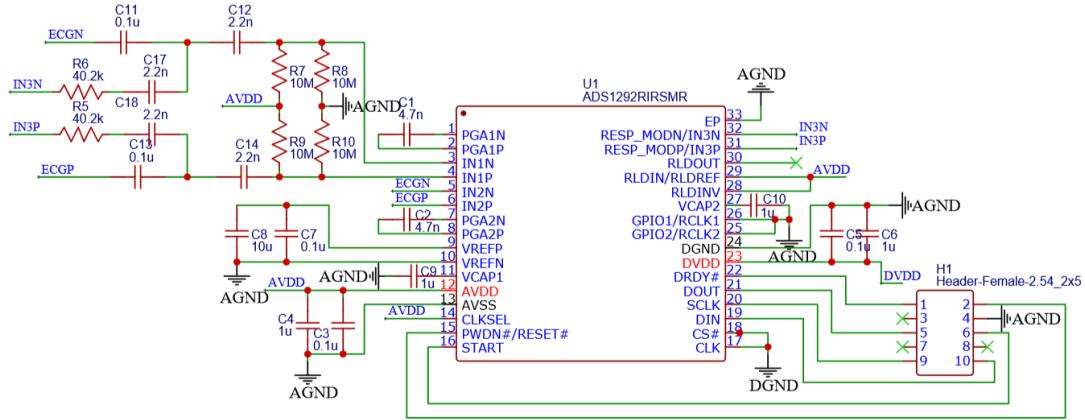


Figure 4. 心电采集电路原理图

心电信号采集模块选用 ADS1292R 芯片。ADS1292R 是 TI 公司推出的用于生物电势测量的低功耗、双通道的 24 位模拟前端，适用于心电图、便携式心电图的诊断和患者检测<sup>[7]</sup>。ADS1292 具有以下特点：内部含一个低噪声可编程控制器（PGA），可编程增益为 1、2、3、4、6、8 以及 12；一个高分辨率（24 位）Delta-Sigma 模数转换器，7 种数据转换速率，由 125SPS 到 8000SPS；输入参考噪声和偏置电流分别为 8uVpp 和 200pA；低功耗，仅 335uW；高共模抑制比，可达 105dB；串行外设接口支持 SPI 通信[6]。ADS1292R 作为一种测量生物电的解决方案，与传统的分离元件测量方案相比，具有以下优点：元器件数量缩减 92%，电路板尺寸缩减 92%，功耗降低 94%。提高了系统的可靠性与患者的移动能力。引脚排列与引脚功能如 Figure 5 与 Figure 6 所示。

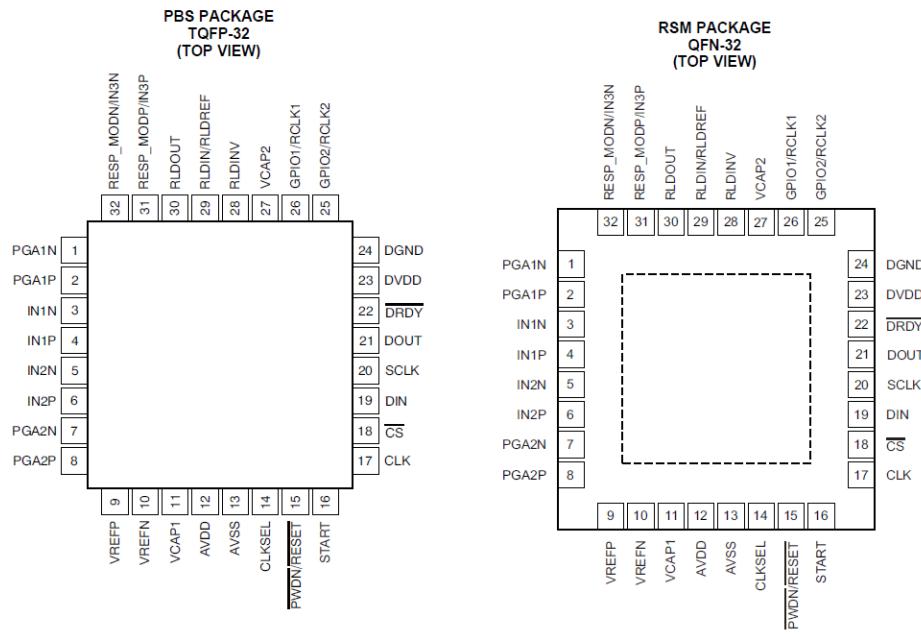


Figure 5. Pin configurations of ADS1292R

NAME	TERMINAL	FUNCTION	DESCRIPTION
AVDD	12	Supply	Analog supply
AVSS	13	Supply	Analog ground
CLK	17	Digital input	Master clock input
CLKSEL	14	Digital input	Master clock select
CS	18	Digital input	Chip select
DGND	24	Supply	Digital ground
DIN	19	Digital input	SPI data in
DOUT	21	Digital output	SPI data out
DRDY	22	Digital output	Data ready; active low
DVDD	23	Supply	Digital power supply
GPIO1/RCLK1	26	Digital input/output	General-purpose I/O 1 or resp clock 1 (ADS1292R)
GPIO2/RCLK2	25	Digital input/output	General-purpose I/O 2 or resp clock 2 (ADS1292R)
IN1N <sup>(1)</sup>	3	Analog input	Differential analog negative input 1
IN1P <sup>(1)</sup>	4	Analog input	Differential analog positive input 1
IN2N <sup>(1)</sup>	5	Analog input	Differential analog negative input 2
IN2P <sup>(1)</sup>	6	Analog input	Differential analog positive input 2
PGA1N	1	Analog output	PGA1 inverting output
PGA1P	2	Analog output	PGA1 noninverting output
PGA2N	7	Analog output	PGA2 inverting output
PGA2P	8	Analog output	PGA2 noninverting output
PWDN/RESET	15	Digital input	Power-down or system reset; active low
RESP_MODN/IN3N <sup>(1)</sup>	32	Analog input/output	N-side respiration excitation signal for respiration or auxiliary input 3N
RESP_MODP/IN3P <sup>(1)</sup>	31	Analog input/output	P-side respiration excitation signal for respiration or auxiliary input 3P
RLDIN/RLDREF	29	Analog input	Right leg drive input to MUX or RLD amplifier noninverting input; connect to AVDD if not used
RLDINV	28	Analog input	Right leg drive inverting input; connect to AVDD if not used
RLDOUT	30	Analog input	Right leg drive output
SCLK	20	Digital input	SPI clock
START	16	Digital input	Start conversion
VCAP1	11	—	Analog bypass capacitor
VCAP2	27	—	Analog bypass capacitor
VREFN	10	Analog input	Negative reference voltage; must be connected to AVSS
VREFP	9	Analog input/output	Positive reference voltage

(1) Connect unused analog inputs to AVDD.

Figure 6. Pin assignments of ADS1292R

ADS1292R 的电路配置参考了 demonstration board 的电路设计[7]、以及使用 ADS1292R 构建心电采集系统的论文[8][9]和博客[10]中的电路设计。我们采取的电路设计如 Figure 7. 所示。采用最简洁的 I 导联方式（即左右手分别接入 ECGN、ECGP），通过简单 RC 滤波后接入 IN1N/P、IN2N/P、IN3N/P，具体电路借鉴 demo board[7]与一篇博客 [10]中的设计。采集心电信号的电极片用两块圆形裸露的焊盘来代替；删减了右腿驱动电路的部分，RLDINV 与 RLDIN 接入高电平，RLDOUT 没有引线接出。AVDD 和 AVSS 分别接入模拟电源和模拟地，DVDD 和 DGND 分别接入数字电源和数字地。Analog supply 为 3.3V，digital supply 为 1.8V。供电和地两个管脚之间接入滤波电容。VCAP1 和 VCAP2 分别接入两个旁路电容。PGA1N 与 PGA1P、PGA2N 与 PGA2P 之间依据 datasheet 的供电电路，接入一个 4.7nF 电容，而 VREFN 与 VREFP 接旁路电容后接地。CLKSEL 为时钟片选管脚，我们使用内部时钟信号，因此该管脚接高电平，CLK 是外部输入时钟，因此接地。CS#是片选信号，用于连接多个芯片时选择不同芯片，我们将其接地即可。GPIO1/2 可当作输入输出管脚，没有用到因此作接地处理。除此之外，DRDY#是 ready 信号，DIN 是 SPI 数据输入，DOUT 是数据输出，SCLK 是 SPI 的时钟，PWDN#/RESET#是 power down 或复位信号，START 是开始信号，这六个信号与 FPGA 通信，因此使用一个直插排母便于连线到开发板。EP 是芯片外壳，作接地处理。具体各管脚的功能可参考 ADS1292R 的 datasheet 中 pin configurations 部分[6]，以及各管脚信号的详解部分。

#### 4. PCB 绘制、焊接与调试

##### (1) PCB 绘制

PCB 的绘制当中，我们需要注意的事情有以下几点：

i、电路元件的选取：

同样的一类电路元件，型号对封装的影响很大。例如升压电路的 XH2.54 端子，不同的型号对应封装不同。XH2.54 端子如图示：

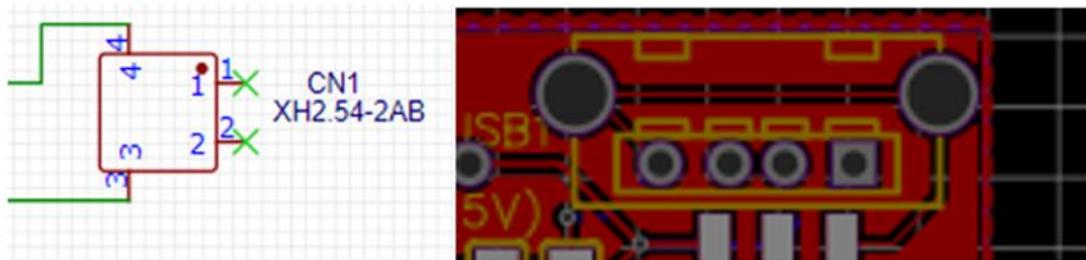


Figure 7. XH2.54 端子原理图符号与 PCB 示意图

最早的时候，选取的 XH 端子对应的封装面积很大，其中的 USB 接口部分就占据了很大的面积，但是更换型号之后，封装变小了，这是因为原先巨大的接口变成垂直于 PCB 板，这样的就节省了 PCB 板的空间资源。

还有升压电路的芯片 TPS61027，如 Figure 8. 升压芯片 TPS61027 原理图符号与 PCB 示意图：

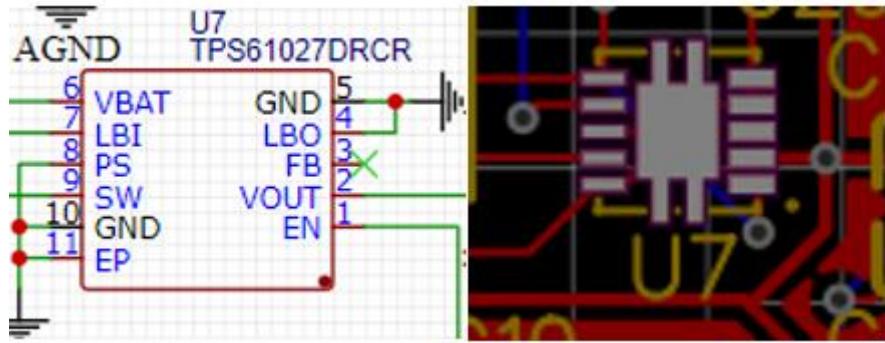


Figure 8. 升压芯片 TPS61027 原理图符号与 PCB 示意图

这个电路原先型号对应的封装很小，接口密集，不适合进行线路连接。我们更换电路的型号，方便接口导线的处理。

#### ii、电路网络设计：

我们的 PCB 板大小是 2400\*2600mil，排除两个半径 400mil 的焊盘，PCB 板的布局资源还是比较紧张的。PCB 分为顶层和底层，根据惯例，底层一般走地线或者高位电压等等信号线，不过这两种一般不在同一层，防止短路。但是我们的 PCB 板只有两层，而且第一层需要布置电路元件，因此将高电压线和地线布置在同一个底层，只能尽量拉开走线。（不过由于设计经验不足，导线通常需要在两层来回穿插，高电压线和地线不能很好的区分）。

Figure 9 是我们的主体芯片 ADS1292R 周围的布线情况，分为顶层和底层：

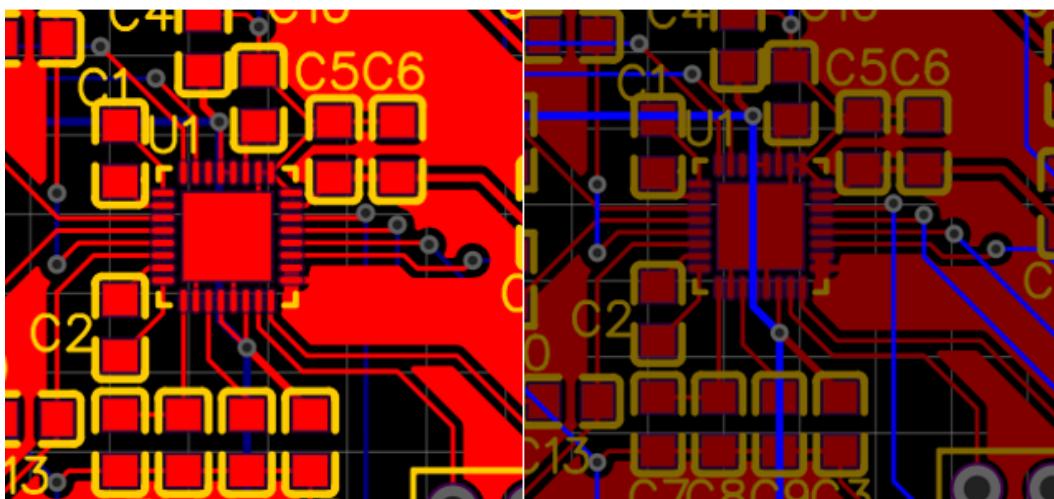


Figure 9. ADS1292R 布线示意图

可以看到，因为电容需要尽量靠近芯片接口的要求，芯片周围堆积了很多的电容元件。原先我们选取的是电解电容，但是在封装上，电解电容比普通电容的封装更大，而且是圆形，无论是结构上还是美观上都无法达到要求。经过商讨和老师建议，我们最终决定选用普通电容，尽可能优化结构。

但这样的结构优化也远远不足，通过图片可以看到，芯片的周围有很多通孔，这是为了保证走线，防止导线交叉而出现 DRC 错误。同时芯片的引脚也过于密集了。对于 ADS1292R，我们选取了这样的芯片也就确定了型号，无法更换封装，因此我们决定将导线变细，将 10mil 改成 6mil。当然，这样做必定会影响导线的传输效率，对于一些比较重要的线路网络，例如地线网络和 5V、1.8V 等网络，我们依旧采取原先的 10mil 导线。

除此之外，还有大面积地铺地、敷铜等措施，这一方面是 PCB 设计要求，另一方面也能帮助我们区分不同的网络，防止它们相互干扰。不过也可以看到，有些部位没有成功敷铜，这是因为该部分线路过于密集导致的，比如 Figure 9 所示芯片周围的网络。

### iii. 采集电极的选取：

我们选取的电极是我们自己绘制的，半径为  $400\text{mil} \approx 1\text{cm}$  的圆形电极，只需要将双手手指放在不同的圆上，芯片就能够通过 ECGP、ECGN 两个接口采集人体信号，包括心电信号。

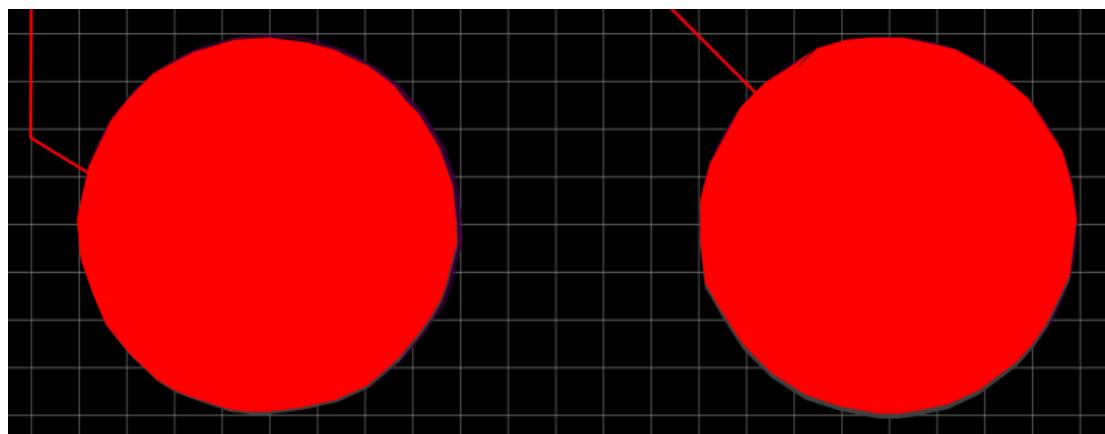


Figure 10. 采集电极 PCB 设计图

如 Figure 10. 采集电极 PCB 设计图，将 PCB 板敷铜区的“可见”更改为“不可见”，但圆形电极上依然有红色标示，这是因为两个电极本身也是敷铜的。同时，两个电极需要进行实心填充，否则就只有丝印层发挥作用，可以看到圆形不是标准的正圆，而是类似多边形，这是实心填充不精细导致的。

### iv. 最终 PCB 绘制示意图与投板预览图展示

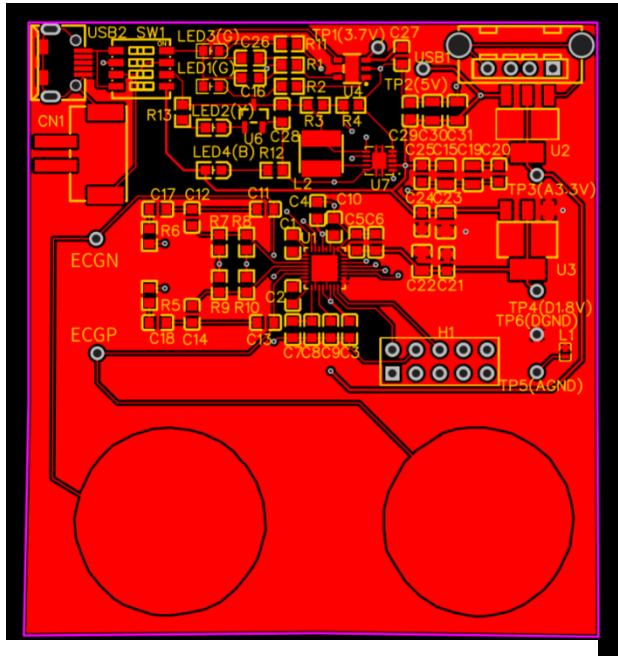


Figure 11. PCB 绘制成品图

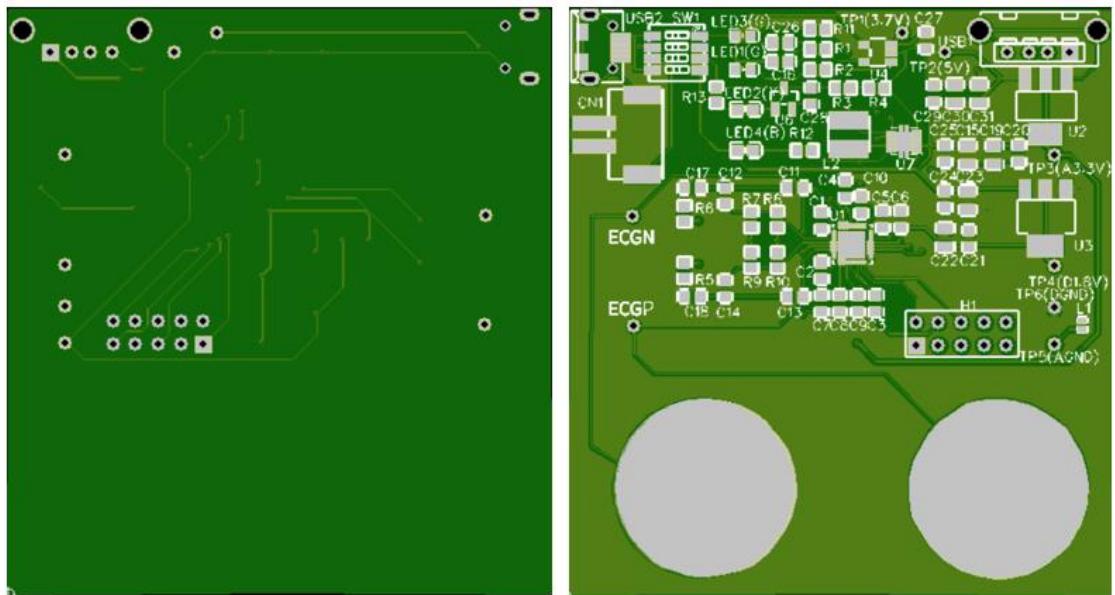


Figure 12. 投板 PCB 预览图

## (2) 焊接+初步调试电压

因为是第一次焊接贴片器件，所以买了几块PCB练习板练手，蓝色那块板是一个流水灯，可以检验焊接的效果，起初有虚焊的情况，检查后可以正常亮灯。

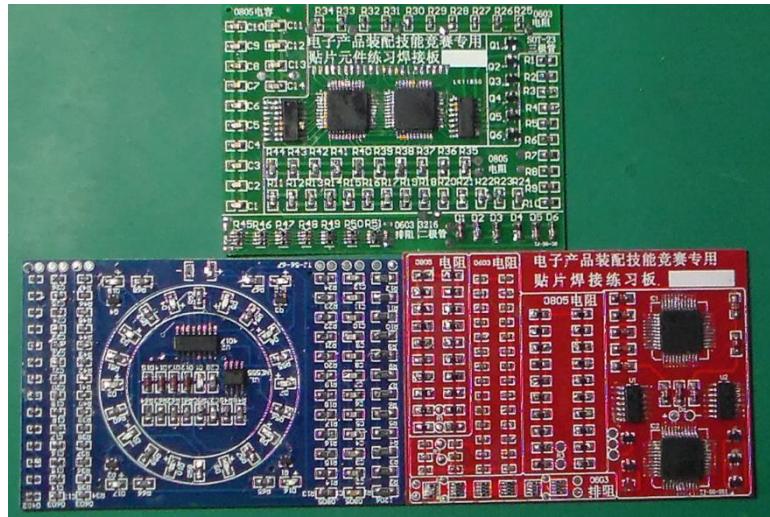


Figure 13. 三块 PCB 练习板

焊接遇到不少问题，焊坏了两块板子，有个是焊盘焊掉了，有个是 QFN 封装的芯片焊坏了。两个 QFN 封装器件很难焊，micro-USB 接口焊进锡没法插接口也没法处理。最后一下将整块电路全部焊接完毕，之后测试发现没有在 TP 点测试期望的电压，直接供电的也存在问题，故焊接失败。

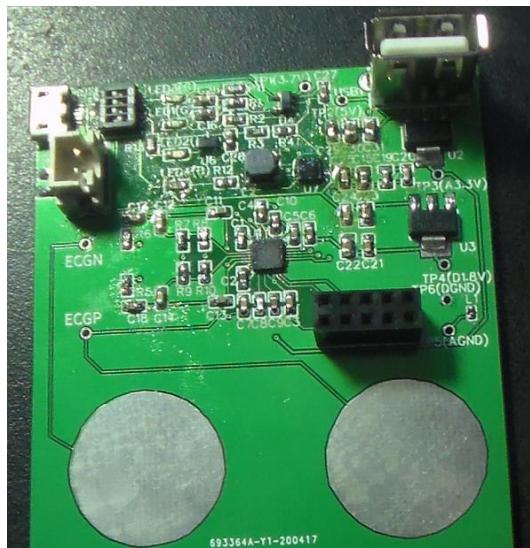


Figure 14. 焊接失败的 PCB 板子

在与老师交流讨论后进行逐个模块焊接，放弃充放电电路，采用直接供电电路，得出的稳压输出是正确的。

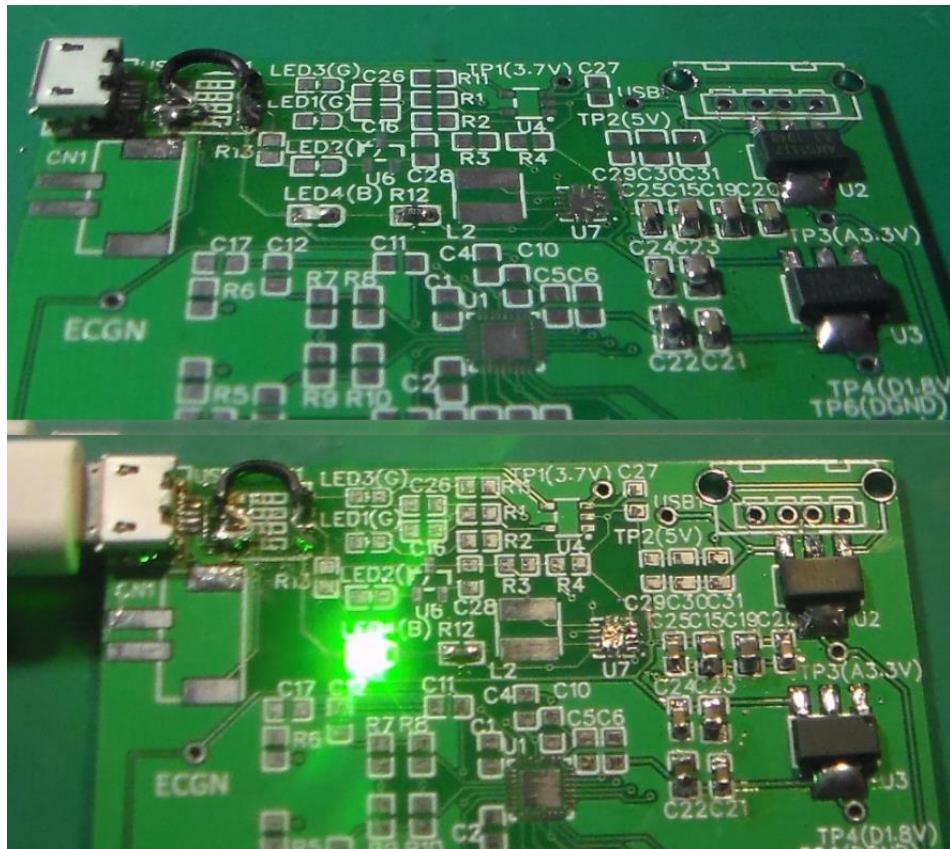


Figure 15. 直接供电电路部分测试

直接供电部分焊接完后，最先焊接最难焊的 ADS1292R 芯片，为 QFN 封装器件。如 Figure 16 有反光，用相机放大检查后引脚无短路情况。

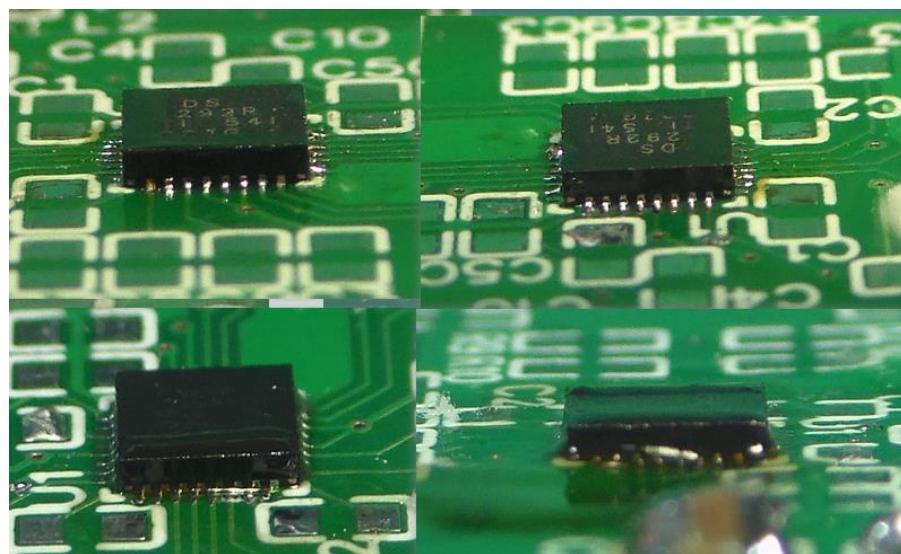


Figure 16. ADS1292R 焊接图

最后完成采集模块的焊接，整个 PCB 板焊接完成。进行初步的电压测试，得到期望的输出电压（1.8V 和 3.3V）。

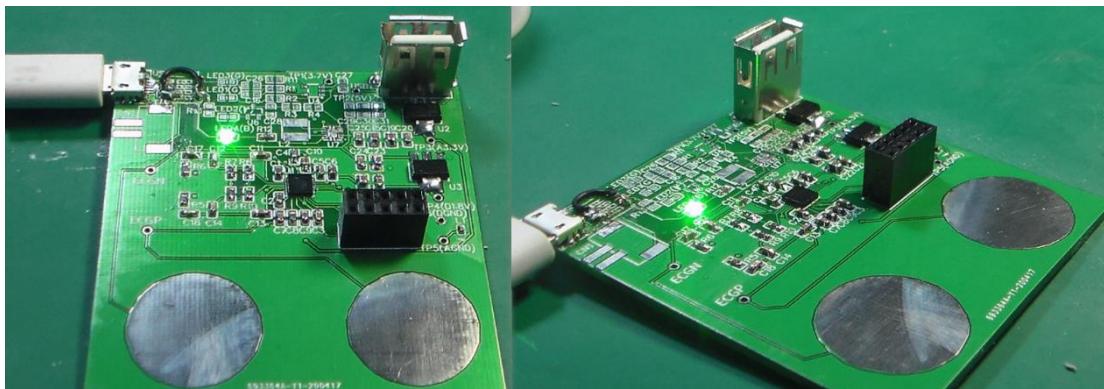


Figure 17. 完整电路焊接图

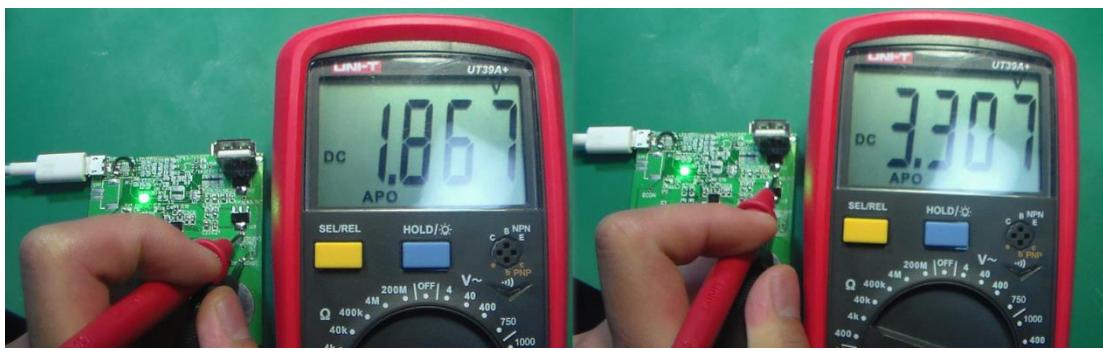


Figure 18. 初步电压测试图

### (3) PCB 调试

调试的工具是示波器，如 Figure 19 所示：

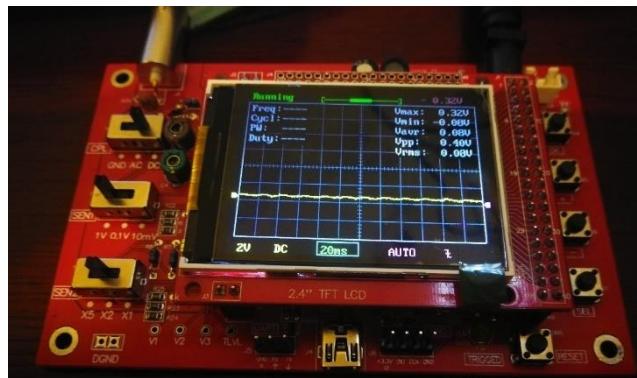


Figure 19. 自行采购的示波器

第一步调试的目的是检测示波器的工作状态和一般情况下的芯片有无信号输出（无滤波情况）。由于我们实验条件限制，我们选取的是市面上购买到的，由 STM32 单片机为基础

设计的示波器。左边的三个开关，从上到下分别控制示波器的耦合方式、量程和倍率。右边的按钮从上到下的功能分别对应：OK 键（负责冻结画面和刷新画面）、+和-键（负责调整被选取到的变量）、SEL 键（选取想要看到的变量，选取到的信号会被高亮）。

其他部分的功能诸如触发条件设置之类的功能暂时用不到，右上角是充电线接口，为示波器供电。左上角是输入信号接口。我们将排针插上芯片采集部分的元件 header-female 排母，示波器红线和黑线分别连接排母的 DOUT（5 号接口）和 AGND（4 号接口），然后给 pcb 芯片供电，观察示波器的信号输出，并调整量程和倍率，获得好的观察效果。

示波器的显示结果如 Figure 20 Figure 21 示，

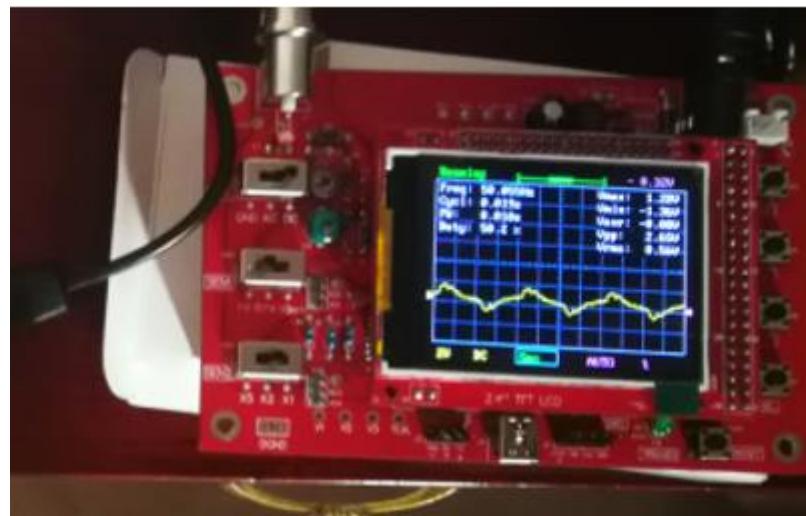


Figure 20. PCB 未供电显示结果

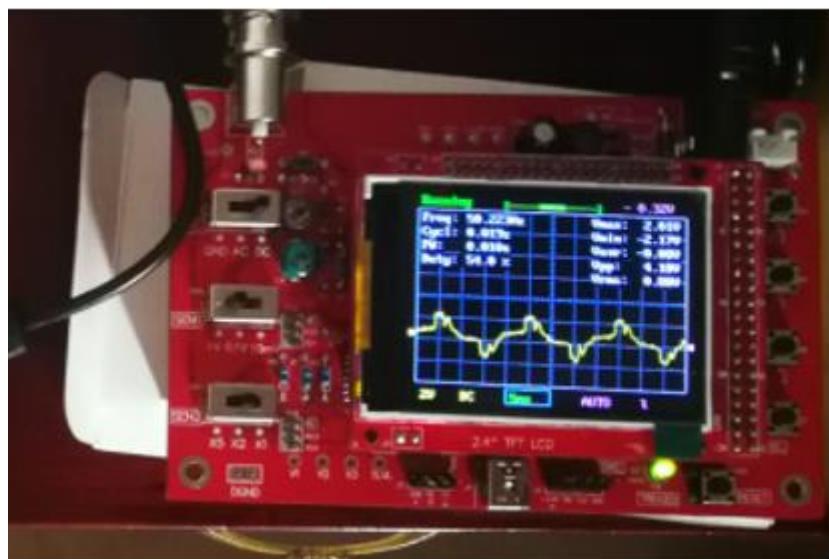


Figure 21. PCB 供电后显示结果

通过对比不同供电情况下，示波器的波形输出状态可以确定芯片的工作情况。未供电的情况下，示波器输出的波形是规律的类三角波（或者说是不同噪声的叠加，但在大体上以 50Hz 的正弦波工频噪声为主）。但是供电之后，示波器的输出有了变化。在噪声的波峰位置出现了扰动，这可能就是芯片工作状态的表现了。如果配置了芯片，芯片就能够进行输出，通过一定的算法，将噪声除去，就可以得到有用的心电信号。

## 四、软件设计与实现

### 1. FPGA 处理

#### (1) SPI 通信

##### i. SPI (Serial Peripheral Interface) 的介绍

ADS1292R 的芯片对应的通信协议是 SPI。SPI 是 Serial Peripheral Interface 的缩写，意思是串行外围接口，是一种高速全双工，同步的通信总线。主要应用于 EEPROM，FLASH，实时时钟，AD 转换器还有数字信号处理器和数字信号解码器之间。SPI 在芯片上大多只占用 4 根线，节约芯片管脚，为 PCB 布局节省空间，提供方便。由于这种特性，越来越多集成度要求较高的芯片采用了这种通信协议，例如 AT91RM9200。

SPI 的主从模式连接如图[11]：

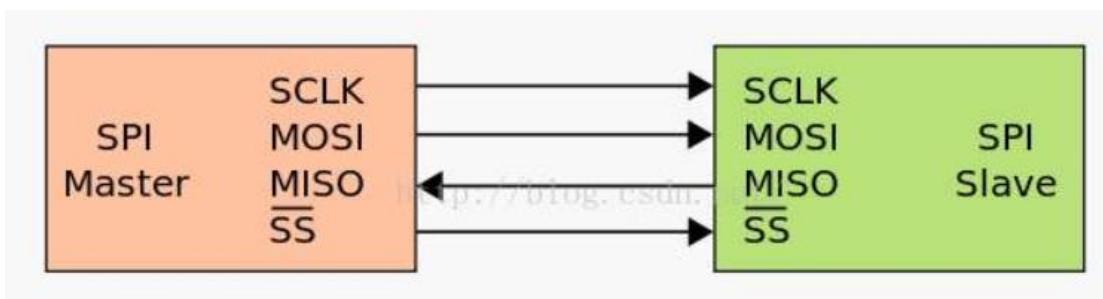


Figure 22. SPI 主从模式

SPI 的工作模式通常是一个主设备和一个或多个从设备，需要至少四根线（如果是单向传输，那么用三根也可以）。基于 SPI 设备共有的，它们分别是 SDI（数据输入）、SDO（数据输出）、SCK（时钟）、CS（片选）信号。其中，SDO 也可以写成 MOSI，对应主设备数据输出，从设备数据输入；SDI 写成 MISO，对应主设备数据输入和从设备数据输出。SCK 是主设备产生的时钟信号，CS 则是主设备产生的从设备使能信号。

SPI 总线的工作状态有四种，分别由两个参数确定：CPOL 和 CPHA，分别是时钟极性选择和时钟相位选择。CPOL 为 0 时，SPI 总线的空闲状态为低电平；CPOL 为 1 时，总线的空闲状态是高电平。CPHA 为 0 时，SCK 在上升沿采样，为 1 时 SCK 在下降沿采样。当 CPHA = 0、CPOL = 0 时，SPI 总线工作于方式一，这也是本项目的 SPI 程序的工作状态。

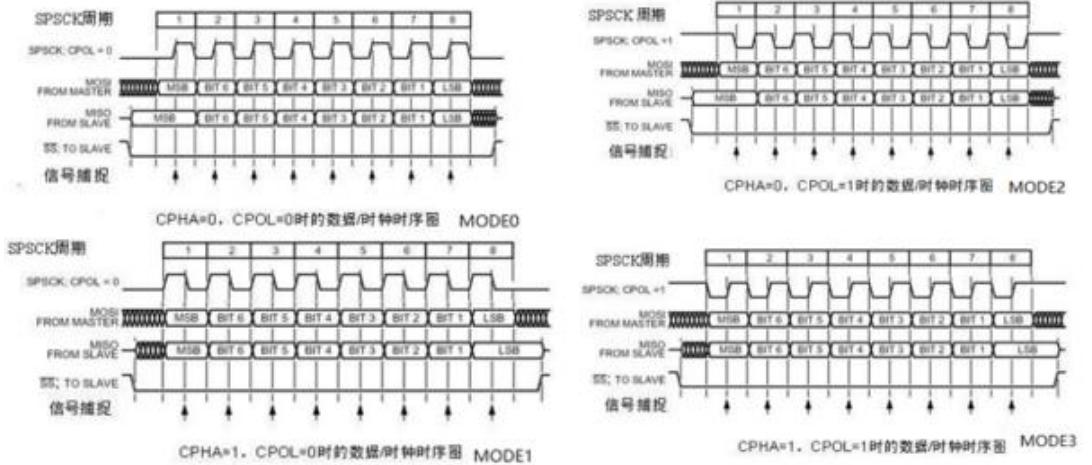


Figure 23. SPI 的四种工作模式波形时序

我们的 ADS1292R 芯片的 SPI 对应管脚如图示：

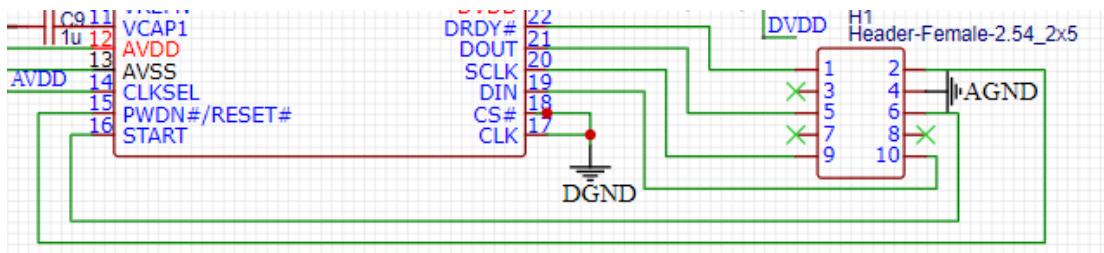


Figure 24. ADS1292R 的 SPI 对应管脚

CS#信号控制片选，但是我们将芯片设定为从设备，那么实际上也不需要这个接口了，CS#固定接地。DIN 是从设备输入，主设备输出，对应 MOSI，对应的信号是 FPGA 控制生成的 register 信号。SCLK 和 DOUT 分别对应 SPI 中的 SCK 和 MISO，分别是时钟和主设备输入、从设备输出。DRDY#是一种比较特殊的使能信号。它的作用是，当 DRDY 置零的时候，会导致 CS#片选信号置 0，随后 DRDY#置 1，表示工作状态为忙碌，然后开始发送和接收数据。不过这是芯片作为主设备有用，在从作为从设备的情况下作用可以忽略[13]。

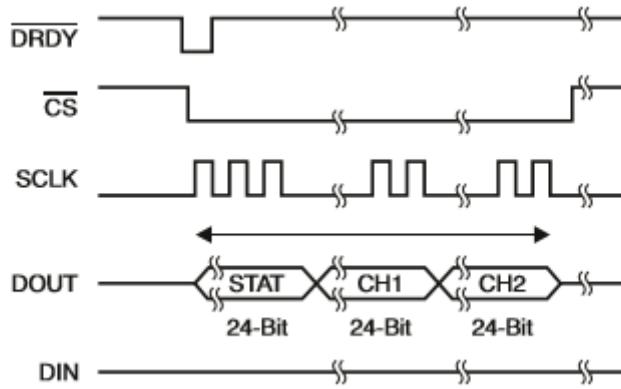


Figure 25. 时序模拟图

根据这些要求，在阅读 ADS1292R 的 datasheet 之后，我们最终确定，主设备为 FPGA、从设备为 PCB 板；SPI 的采样时钟为 2kSPS，也就是说 SCLK 的频率为 2kHz；采样数据的对应寄存器的位宽为 24。

#### i i. SPI 部分代码编写

在 Verilog 代码编写中，主要有状态机 SPI、波特时钟 Baudtime、锁存模块 lock、寄存器配置模块 register 这几个模块。

首先在名为 SPI 的模块定义了一个 24 位的状态机（因为寄存器的位宽是 24，不过实际上和 8 位的状态机在代码实现上没有区别）。和 UART 协议类似，需要通过状态机将芯片输出的 24 位数据从高位到低位，由 I\_spi\_miso 变量一位一位地输入，存入 24 位寄存器 oData 当中。同时，又定义了一个表示状态机状态的寄存器变量 R\_rx\_state，因为是从 0 数到 23，数据位宽用 6 位。根据 R\_rx\_data 的变化指示 oData 存储的位置。在 oData 数据读入完成后，将 oDone 信号置 1，表示读入完成。将数据输出的情况也同理（当然，对芯片来说不需要数据重新输出到芯片中，我编写输出的部分是为了用作串口测试）。输出就是将已经从 oData 得到数据的 iData 中的数据从高位到低位，在状态寄存器 R\_tx\_state 的指示下，传入 O\_spi\_mosi 中，并在完成后将 iDone 的电位置 1。

波特时钟 Baudtime 模块用于生成 SCLK 信号。根据要求，SCLK 的频率是 2kHz。使用用 EG01 板卡的系统时钟，100MHz 的 sysclk 进行分频，每 12500 个 sysclk 周期后，clk 的信号改变一次；clk 信号改变 2 次，sck 时钟改变一次，这样获得到 SCK 信号的频率即为 2kHz。

锁存模块 lock 是用于将 oData 信号进行锁存的模块。因为 SPI 通信的时候，寄存器 oData 是不断改变的，为了防止误码，需要在每次 24 位数据完全存入的时候，将数据锁存到一个中间的寄存器变量中。这样做有三个好处：首先，锁存后的信号不会频繁改变，防

止因为信号不断传输和跳变导致的误码情况；其次，锁存的信号可以留出一个接口，添加滤波模块，这样做可以支持滤波模块对全部的 24 位数据进行处理，而不是一位一位地处理，提高了效率和准确性；最后，锁存得到的信号可以直接传给 iData 进行发送，也可以用显示模块显示在 FPGA 上，观察信号有没有进入 FPGA。

ADS1292R 的芯片并不是一通电就可以开始传输数据的，在传输数据之前还需要对其内部的寄存器进行配置。(因此调试部分需要利用 fpga 的寄存器配置模块，无法直接获得数据，上面的调试部分也仅仅能够知道芯片处在工作状态罢了)。根据 ADS1292R 芯片的 data sheet，我们确认配置信号为 00000100\_101111000\_01100000\_01100000 为 4 个 8 位数组的输出方式，分别对应配置芯片的四个寄存器状态 config1、config2、ch1set 和 ch2set [6]。

因为无法使用 24 位状态机，选择将四个 8 位数组写成 1 个 32 位的数组，用 SCLK 时钟作为触发一位位地输入芯片。不使用状态机的原因是芯片的配置并不是持续的，只要输入了这 32 位数据，芯片就会开始工作，那么该接口就可以置空了，因此不需要状态机不断运作，只需要一次性完成传输即可。实际上，对应 0\_spi\_mosi2 的信号才是对从设备芯片的 mosi 信号；0\_spi\_mosi 则是用于串口测试的信号。具体代码附在附件的压缩包里。

### iii. SPI 工程的仿真与测试

在 testbench 中写了一个随机的，频率为 2kHz 的信号 I\_spi\_miso 进行仿真，得到结果如图所示。0\_spi\_sck 在 10ms 内有 20 个周期，表示 2kHz 的频率。前 20ms，0\_spi\_mosi 没有输出，这是因为需要 0\_spi\_mosi2 信号开启寄存器，配置寄存器状态，等到 32 位数据完全输入芯片后，才会开始工作。oData 每次完成一次读取，都会生成 oDone 信号；相应的 iData 和 iDone 也是同理。

通过 Figure 26 可以看出，测试结果符合 spi 的运作机制。



Figure 26. SPI 工程仿真结果

测试用逻辑分析仪如 Figure 27 示：



Figure 27. 逻辑分析仪

逻辑分析仪对应软件的界面如 Figure 28 示：

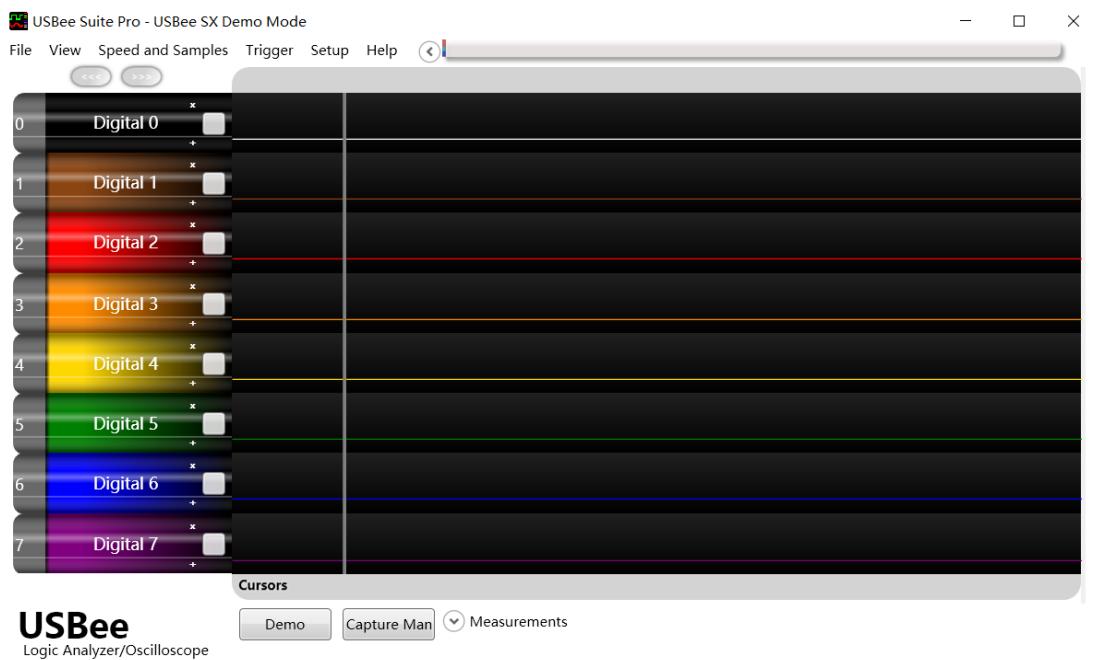


Figure 28. 逻辑分析仪测试软件界面

测试的方法是利用逻辑仪的分析能力，将逻辑仪的接口用导线连接 FPGA 对应的输出接口，然后利用串口调试助手尝试发送数据，对应在 FPGA 上的输入输出就会反应在逻辑仪的界面当中。SPI 工程的 xdc 文件部分配置如 Figure 29 所示。

```

set_property -dict {PACKAGE_PIN P17 IOSTANDARD LVCMOS33} [get_ports sysclk ]
set_property -dict {PACKAGE_PIN H17 IOSTANDARD LVCMOS33} [get_ports cnt ]
set_property -dict {PACKAGE_PIN H16 IOSTANDARD LVCMOS33} [get_ports tbsignal ]
set_property -dict {PACKAGE_PIN F13 IOSTANDARD LVCMOS33} [get_ports I_spi_miso ]
set_property -dict {PACKAGE_PIN R11 IOSTANDARD LVCMOS33} [get_ports rst ]
set_property -dict {PACKAGE_PIN P5 IOSTANDARD LVCMOS33} [get_ports I_rx_en ]
set_property -dict {PACKAGE_PIN P4 IOSTANDARD LVCMOS33} [get_ports I_tx_en ]
set_property -dict {PACKAGE_PIN B16 IOSTANDARD LVCMOS33} [get_ports start ]

set_property -dict {PACKAGE_PIN D15 IOSTANDARD LVCMOS33} [get_ports I_spi_miso ]
set_property -dict {PACKAGE_PIN E15 IOSTANDARD LVCMOS33} [get_ports O_spi_mosi ]
set_property -dict {PACKAGE_PIN N5 IOSTANDARD LVCMOS33} [get_ports I_spi_miso ]
set_property -dict {PACKAGE_PIN N2 IOSTANDARD LVCMOS33} [get_ports O_spi_mosi ]
set_property -dict {PACKAGE_PIN F13 IOSTANDARD LVCMOS33} [get_ports O_spi_mosi2 ]
set_property -dict {PACKAGE_PIN A15 IOSTANDARD LVCMOS33} [get_ports O_spi_sck ]

```

Figure 29. SPI 工程的部分管脚配置

SPI 测试出现的问题在于，虽然通过 xdc 文件将输入输出变量多引出几个在对应的排针上，但是串口助手却不能很好地适应 SPI 传输要求。虽然串口尝试发送数据，但 FPGA 却无法将数据采集下来，输入 FPGA 中，这导致了 SPI 测试无法继续进行下去。

## (2) 数字滤波

在采集人体心电信号的过程中，各种干扰也会混杂其中，主要包括工频噪声、基线漂移和肌电噪声这三种。由于心电信号本身很微弱，上述的干扰噪声将严重影响后续的波形检测以及心律失常判别，而当有很严重的噪声时会使得心电信号被完全淹没。所以，为了使得噪声的影响得以避免，需设计各种合适的滤波器。本项目分别针对心电信号中混杂的这几种干扰，设计了可靠的滤波器使其影响得以消除 [14]。

滤波器设计分为 matlab 仿真、vivado 仿真、上板测试三步。

MIT-BIH 心率失常数据库中的数据采集于 1975–1979 年之间，由 60% 的住院病人和 40% 门诊病人的心电数据组成。MIT-BIH 心电数据库中，导联信号质量在一定范围内变化。每个记录是从 24 小时的心电记录中选取了具有代表性的 30 多分钟的数据片段组成数据库 [15]。该数据库共有 48 个病例，每个病例数据长 30 分钟，总计约有 11600 多个心拍，包含各种正常心拍和各种异常心拍，并且由医学专家对每个心拍作出识别和标注。在美国常被用作心电仪器的检验标准。

MIT-BIH 心率失常数据库中的每个记录都由两个导联信号组成，信号的采样率为 360Hz，采样精度为 12 位 [16]。

在本项目中我们使用 MIT-BIH 心律失常数据库 [17] 作为仿真信号输入。

### i. 基线漂移滤波器

基线漂移是由于人体呼吸、电极移动[8]等产生，一般在 0.5Hz 以下。直流偏移电势是一种在生物电测量中常见的噪声，与基线漂移同属低频噪声信号，因此一并作为基线漂移去除。一般有中值滤波法、三次样条插值法、FIR/IIR 高通滤波器法等。考虑到实际情况对于精度要求并不高，所以在本项目中我们采用 3Hz 作为阈值，设计 FIR 高通滤波器。

### ii. 肌电噪声滤波器

人体肌肉的颤动造成了肌电噪声，其频率一般在 100Hz 以上，对心电信号来说是一种高频噪声，低通滤波器的设计能够将其滤除[14]。综合上述两种噪声，构造一个 3Hz–100Hz 的带通滤波器即可以实现对心电信号的预处理。

使用 matlab 的滤波器设计工具得到带通滤波器归一化后的参数如下所示

```
hn =  
列 1 至 11  
0 1178 3703 4560 -1240 -20257 -59536 -124708 -217304 -331678 -452224  
列 12 至 22  
-551786 -592148 -527230 -309084 103881 737683 1594303 2645868 3833108 5069046 6247910  
列 23 至 33  
7258175 7997731 8388607 8388607 7997731 7258175 6247910 5069046 3833108 2645868 1594303  
列 34 至 44  
737683 103881 -309084 -527230 -592148 -551786 -452224 -331678 -217304 -124708 -59536  
列 45 至 50  
-20257 -1240 4560 3703 1178 0
```

Figure 30. 带通滤波器归一化参数

先采用 matlab 进行仿真，确认滤波器效果并生成上板测试的数据。

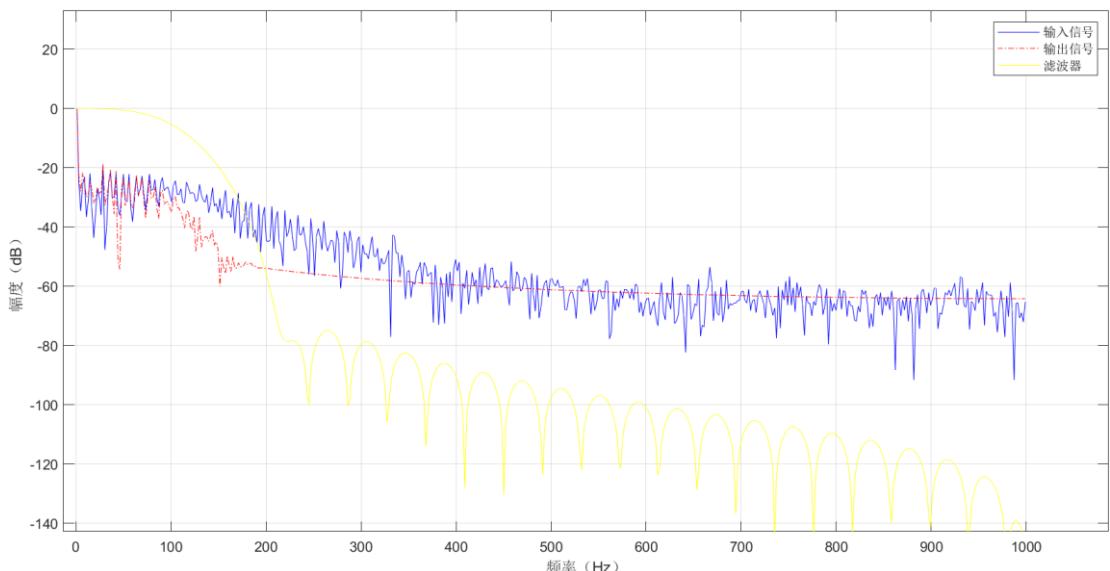


Figure 31. matlab 仿真滤波

从 Figure 31 幅频响应中可以看出，信号 100Hz 以上的信号得到有效的遏制，并在 50Hz 频段实现了陷波。作为对比在下路加了一个全频段的噪声来显示对于高频的抑制效果。加噪声后可以从看出抑制高频效果也十分出色。

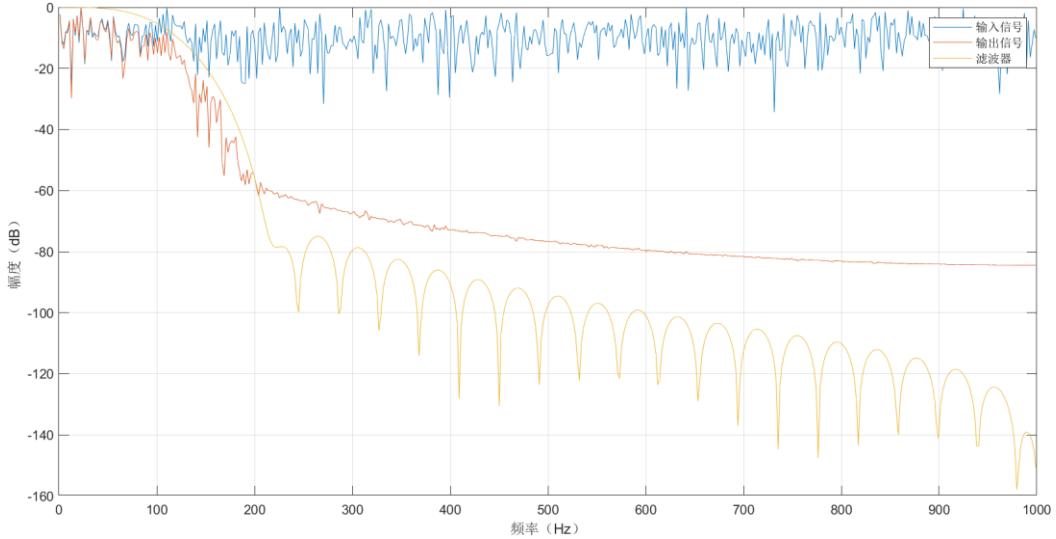


Figure 32. 加入噪声的滤波仿真

Figure 33 是原始信号的波形图与滤波后心电信号。由于噪声的影响，原始数据波形毛刺多，并且在纵轴上存在漂移。希望通过滤波滤除掉高频分量和低频直流噪声来使图像得到修正。

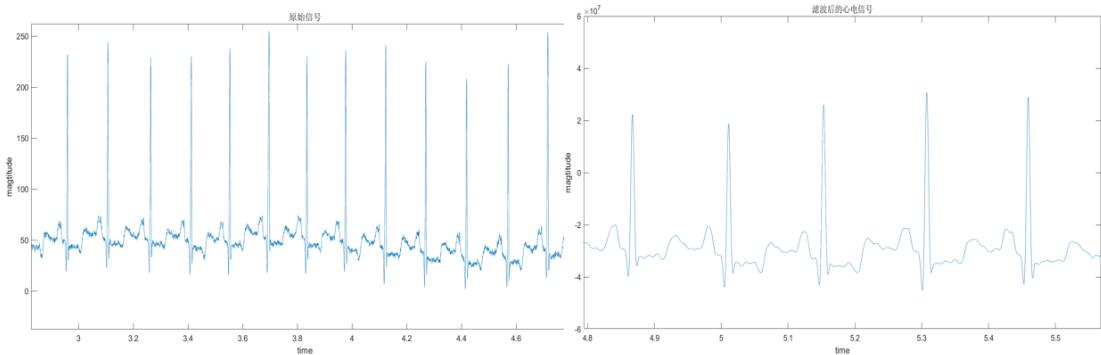


Figure 33. 原始心电信号与滤波后心电信号

滤波后效果如右图所示，可见 QS 波的毛刺得到消除，图像平滑。并且在纵轴上的直流分量得到滤除。接下来将上板，利用 FPGA 实现滤波效果。由于 ADS1292R 送来的是一个 24 位的符号数，因此在实现数字滤波的时候需要注意几个细节，首先可以通过相同系数相加来减少一步计算量，在这里需要补充一位来顺势计算符号位。之后就是利用乘法器实现差分方程，在输出后取高 24 位即可。

使用 matlab 生成的二进制数据发送给 FPGA，然后经过 FPGA 实现 FIR 带通滤波器，最后输出回上位机并显示。用 matlab 来调取 FPGA 输出的数据与 matlab 仿真结果对比。

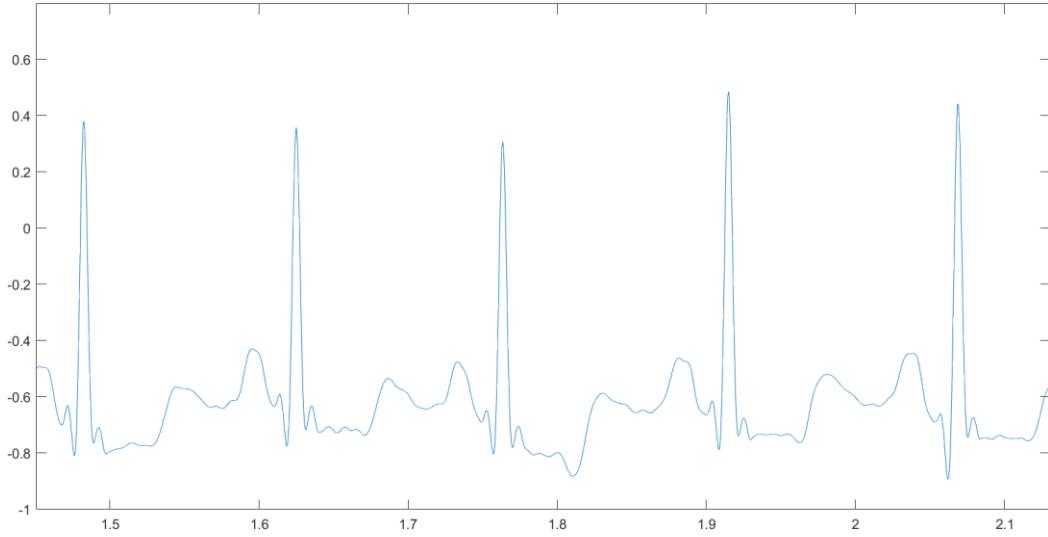


Figure 34. FPGA 滤波结果

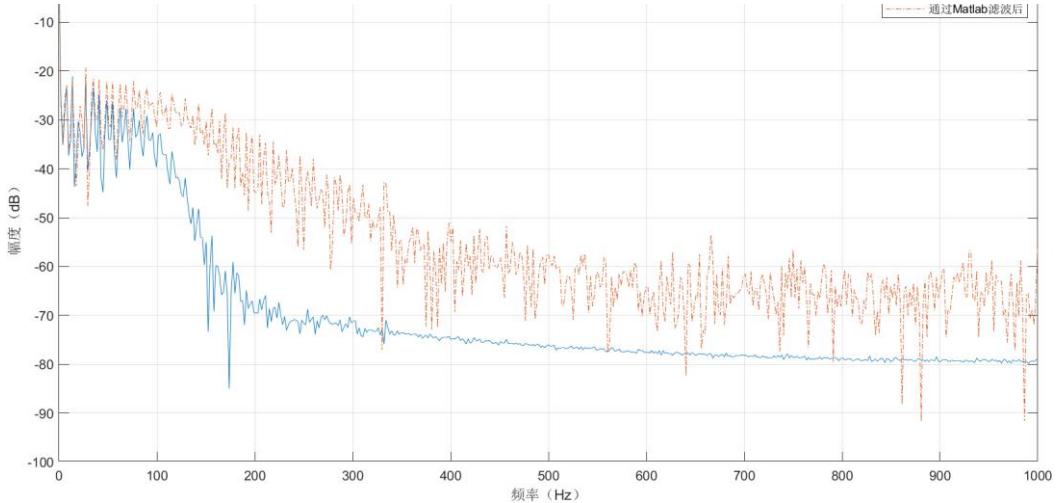


Figure 35. FPGA 仿真与 matlab 仿真对比

FPGA 滤波后和 matlab 的滤波二者幅频响应基本相同，但是存在一个向上的偏移，可能是因为在 FPGA 中为了更好的实现滤波效果修改了一些系数导致的。

### iii. 工频噪声陷波器

工频噪声是由于市电产生的电磁场通过人体和电子设备产生的。硬件上可以通过右腿驱动电路等加以抑制，本项目原理图设计中没有采用右腿驱动电路，而且由于线路存在不对称性，在心电信号处理中去除工频干扰是非常必要的 [14]。

由于使用的 MIT-BIH 心律失常数据库不是国内的，所以仿真实验中的工频频率实际为 60Hz。而在国内，工频噪声一般是 50Hz。在处理心电信号中的工频噪声时，本项目采用了

IIR 滤波器，并设计了双零点双极点的传递函数，其中双零点为 $z = e^{\pm j\omega_0}$ ，双极点为 $z = re^{\pm j\omega_0}$ ，通过陷波从心电信号中滤除工频噪声。工频噪声滤波器的传递函数如下式所示：

$$H(z) = \frac{(z - e^{j\omega_0})(z - e^{-j\omega_0})}{(z - re^{j\omega_0})(z - re^{-j\omega_0})}$$

$$H(\omega) = \frac{e^{2j\omega} - 2se^{j\omega} + 1}{e^{2j\omega} - 2rse^{j\omega} + r^2}$$

$$z = e^{j\omega}, s = \cos\omega_0, \omega_0 = 2\pi \frac{f_0}{f_s}$$

工频噪声滤波器的差分方程为[14]：

$$y(n) = 2rsy(n-1) - r^2y(n-2) + x(n) - 2sx(n-1) + x(n-2), n \geq 3$$

其中， $f_s$  为数据采样频率， $f_0$  为工频频率， $r$  为小于 1 的正数， $r$  越接近 1，则阻带越窄。 $r$  的取值应使工频信号邻域内改变的尽量大，其他频带信号改变的尽量小。在心电信号的频谱 59.5Hz – 60.5Hz 的范围求均方差  $MSE1$ ，表示滤波器对该频段的滤波效果， $MSE1$  的值越大越好；再对其他频带求均方差  $MSE2$ ，表示滤波器对该频段造成的影响， $MSE2$  的值越小越好。 $MSE$  表达式如下

$$MSE = \frac{1}{N} \sum_{k=0}^{N-1} (|Y(k) - X(K)|)^2$$

此处我们直接引用 52[14] 的研究结果，取  $r = 0.99$ 。

matlab 仿真陷波器的幅频和相频特性曲线如 Figure 36 所示，代码名为 filter\_curve.m，具体见附录文件。其中 MIT-BIH 数据库的采样频率为 360Hz，工频噪声为 60Hz。

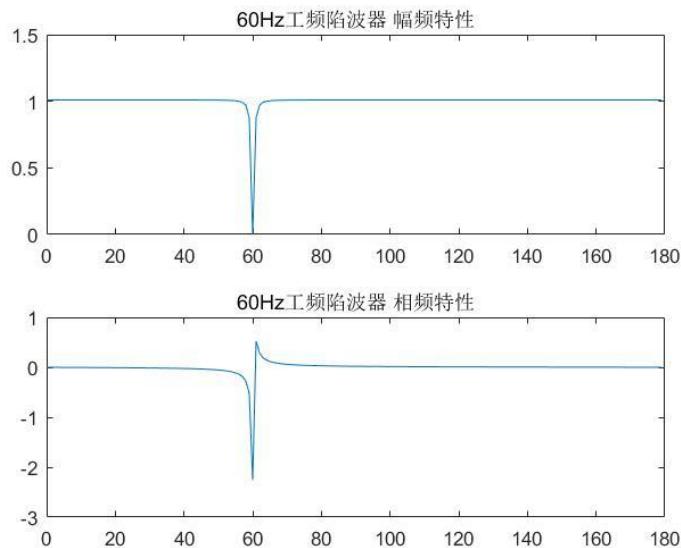


Figure 36. 60Hz 工频陷波器幅频特性与相频特性

从 Figure 36 中可以看到陷波器幅频曲线在 60Hz 处几乎降为 0，而在其他频率处为 1。符合我们对陷波器的要求。

由于实际去除工频干扰时，频率为 50Hz，且数据库采样率与实际数据相差较大，因此补充了使用实际 ADC 转换速率和实际工频噪声频率的相应曲线，在 50Hz 处局部放大后如 Figure 37 所示。实验中采用 2000SPS 采样速率。

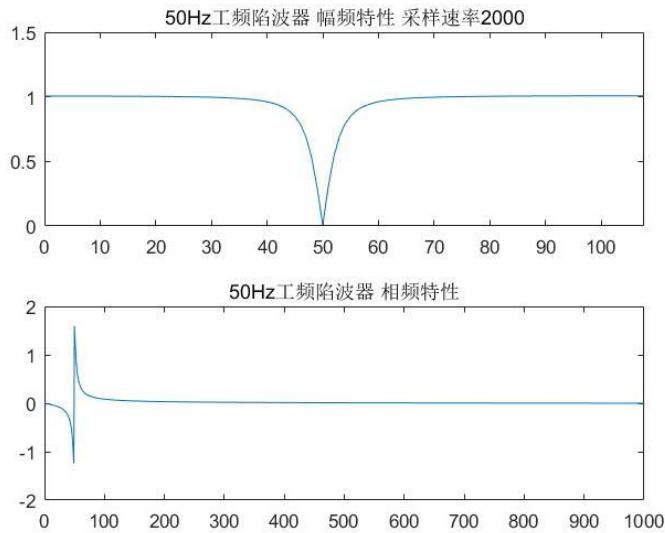


Figure 37. 50Hz 陷波器局部放大，采样速率为 2000SPS

项目使用数据库的心电信号文件，以及数据库提供的 rddata.m 读取心电文件。matlab 仿真使用两种方法，一是使用滤波器参数生成滤波器后对数据滤波，二是使用差分方程对数据直接计算。对比两种方法计算结果，可以发现两种方法得到的结果相似。以此证明了差分方程处理对数据精确度的可行性。除此之外生成滤波前后数据的频谱，以观察滤波器去除工频干扰的效果。频谱图如 Figure 38 所示。滤波器滤波仿真代码名为 noth\_sim.m，运行之前需要先运行 rddata.m 以得到心电数据。

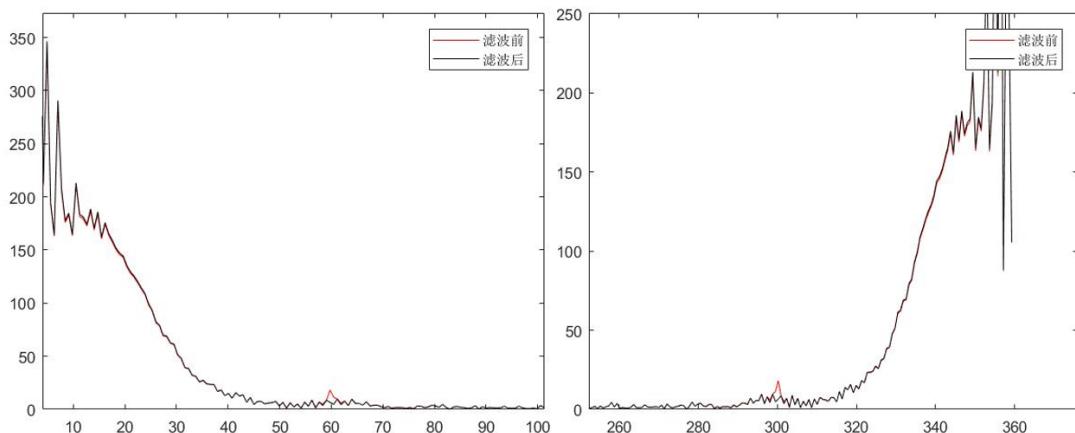


Figure 38. 两张通过陷波器前后的信号频谱（局部）

由于数据只进行了单纯的工频滤波，因此直流分量比较大，但仍可观察到 60Hz 处频谱有明显削弱。除此之外，在 60Hz 的倍频处也有相应削弱，见 Figure 38 右图

在 matlab 里将数据库的心电数据存为 txt 文件，在 vivado 的 testbench 文件中调用数据，可以得到 vivado 仿真结果。vivado 仿真可以得到滤波后的数据，与 matlab 得到的数据对比后可以得知，vivado 仿真证明了陷波器差分方程的有效性。

#### iv. 归一化处理

归一化的处理思路是将每个数据减去最小数据，再除以最大数据与最小数据之差，由此得到所有数据都是 0-1 之间的数字，再乘以 255 即可将数据转为 0-255 之间的数据。实际操作时由于数字滤波模块传出的数据含小数点，因此对分母做了左移位处理以解决该问题。代码见附件的 data2uart.v。

#### (3) 蓝牙通信

蓝牙通信的设计分为两部分，第一是 FPGA 蓝牙模块的功能验证，第二是实现从数字滤波模块向小程序发送数据。首先测试了蓝牙例程，使用手机 APP（BLE 调试助手）测试蓝牙模块，发送命令以点亮数码管。随后阅读并理解蓝牙例程源码，在含 FIFO 的 UART 串口收发器的基础上，配置五个拨码开关，将拨码开关的值赋给蓝牙模块的控制管脚，以此达到控制蓝牙模块的目的；此外，将数据接收的管脚配置到 UART 串口，发送的管脚配置到蓝牙模块的发送管脚，以此达到电脑端的串口助手发送数据到 FPGA，FPGA 通过蓝牙发送数据。配合小程序可以将串口发送的数据通过蓝牙传输给小程序，发送与接收过程记录了视频，附在附录文件中。Figure 1. 展示了蓝牙测试的照片，发送 67，左边两个数码管显示发送数据为 67；右边两个数码管显示接收数据，同样是 67。小程序端同样接收到数据为 67.



Figure 39. 蓝牙测试工程的测试结果

第二步是实现数字滤波模块到小程序模块的传输。这一步主要解决了三个问题。一是数字滤波模块得到的数据远远超过 8 位，而蓝牙使用 UART 传输，一次最多发 8 个 bit。本项目采用的解决方法是将数据归一化到 0-255，即 8bit 的二进制数，以供蓝牙传输，传输一次即为一个数据点。这部分与数字滤波模块衔接，放到数字滤波模块一并处理。第二个问题是数字滤波模块传输的数据是以数组为单位传输，而之前所做的蓝牙模块 demo 工程是以 01 序列传入，rx\_status 的产生成为难题。这一问题的解决是通过在 demo 各模块前加入一个 data\_send 模块，将数组先转换为 01 序列。事实上含 FIFO 的串口收发器是之前写过的，修改 uart\_receiver 模块出现了较多问题，故添加了新模块以解决问题，保留了原工程的相对完整。第三个问题是数字滤波模块的传输速率与蓝牙的发送速率不相同。蓝牙波特率为 9600SPS，而 ADS1292R 传出数字的波特率为 2000SPS。这个问题的解决方法是在 controller 这个模块，使用两个使能信号进行数据间的同步通信。当没有新数据输入时，重复发送当前数据。

最后蓝牙部分的工程分为 clk\_16、clk\_2000、data\_send、uart\_receiver、controller 和 uart\_sender，以及数码管显示模块 scan\_led\_hex\_disp。其中 clk\_2000 用于产生与数字滤波模块同步的时钟信号，数据传输速率为 2000SPS。而 clk\_16 是用于蓝牙通信的时钟，是 9600 传输速率的 16 分频，16 分频的目的是让接收模块收到持续低电平信号，才开始接收 UART 数据，防止噪声干扰带来暂时低电平产生误接收。

data\_send 用于将输入的 8bit 二进制数转换为 01 序列，其作用与 uart\_sender 类似，都是通过状态机，将数组转换为符合 UART 通信协议的 01 序列。uart\_receiver 则将产生的 01 序列存入数组，并在数组的开头产生一个 rx\_status 信号以表明数据状态。controller 利用 receiver 和 sender 产生的使能信号进行数据同步，把 receiver 的数据发送给 sender，由于蓝牙发送速率比接收速率快，因此没有新数据送进来时，设置蓝牙持续发送当前数据。uart\_sender 则将 controller 传送的数组以 01 序列的形式，按 UART 传送协议发送到蓝牙模块的管脚。工程通过外接 5 个 sw\_pin，将这 5 个开关的数值赋值给蓝牙模块的管脚，以此来实现外部开关对蓝牙模块的控制。连接蓝牙时五个开关从左到右（即板卡丝印对应的 SW7-P5 至 SW3-R2）分别为高低高高高，注意丝印对应 SW0-R1 的拨码开关为复位开关，设为高时无法使用蓝牙接收数据。蓝牙等待连接时，蓝色 LED 灯 D17 慢闪；成功连接后，D17 常亮。

RTL 综合得到工程框图如 Figure 40 所示。

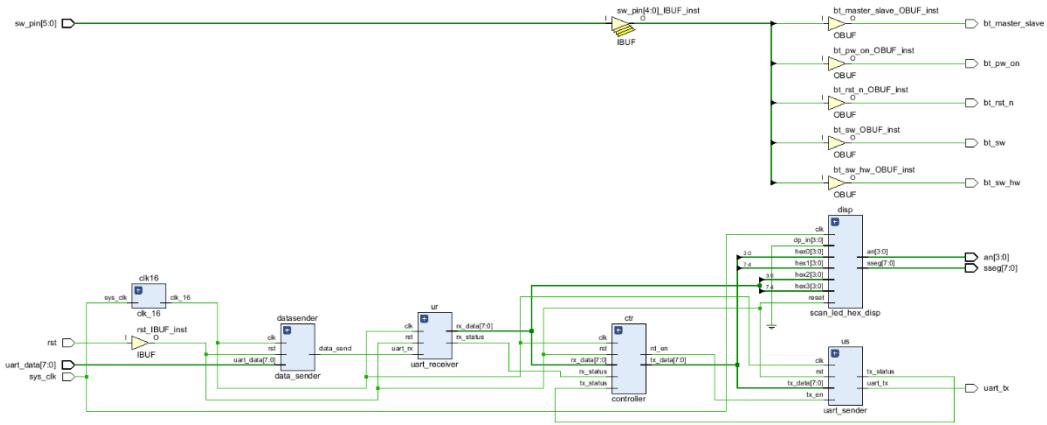


Figure 40. 蓝牙工程 schematic

仿真得到的时序图如 Figure 41 所示，clk 即为板卡自带的 100M 时钟，图上每个周期为 10ns，testbench 每 12ms 传输一个 8bit 数据，如图所有模块都使用 9600 波特率的 16 分频时钟，因此数组传输的长度是相同的，都为 12ms 一个数据。可以求得 uart\_tx 理论上每个数据应发送

$$\frac{9600/10}{2000/24} = 11.52$$

个 01 序列，其中 9600/10 表示蓝牙以 9600 的波特率传输数据，每 10 个 bit 代表一个数据，1 位起始位、8 位数据位以及 1 位停止位。2000/24 表示 SPI 传输原始数据波特率为 2000，每个数据 24 位。



Figure 41. 蓝牙工程仿真结果

查看最下方 tx\_status 可以得到一个数据 6e (即 01101110) 持续的 12ms 内，蓝牙发送了 11 次与 6e 对应的 01 序列 (从 3.34475ms 到 15.96747ms)，与理论结果相符合，即工程完成预期功能。

## 2. 小程序

小程序首先设计了页面布局，分为主页面、蓝牙页面和显示页面。然后实现主要功能——蓝牙通信和动态显示。最后是小程序的整合与调试。

### (1) 页面布局

起初设计是将所有功能放到一个界面，很拥挤也不方便共同开发，所以简化界面，分为主界面、连接设备界面和心电图界面，如 Figure 42。主页是两个按钮，可以分别跳转到两个界面，也可以点击下面的 tabbar 切换界面；连接设备界面主要是蓝牙连接设备并接收数据；心电图界面实现动态显示接收数据，并显示心率计算结果；两个界面的数据传递用全局变量来实现。

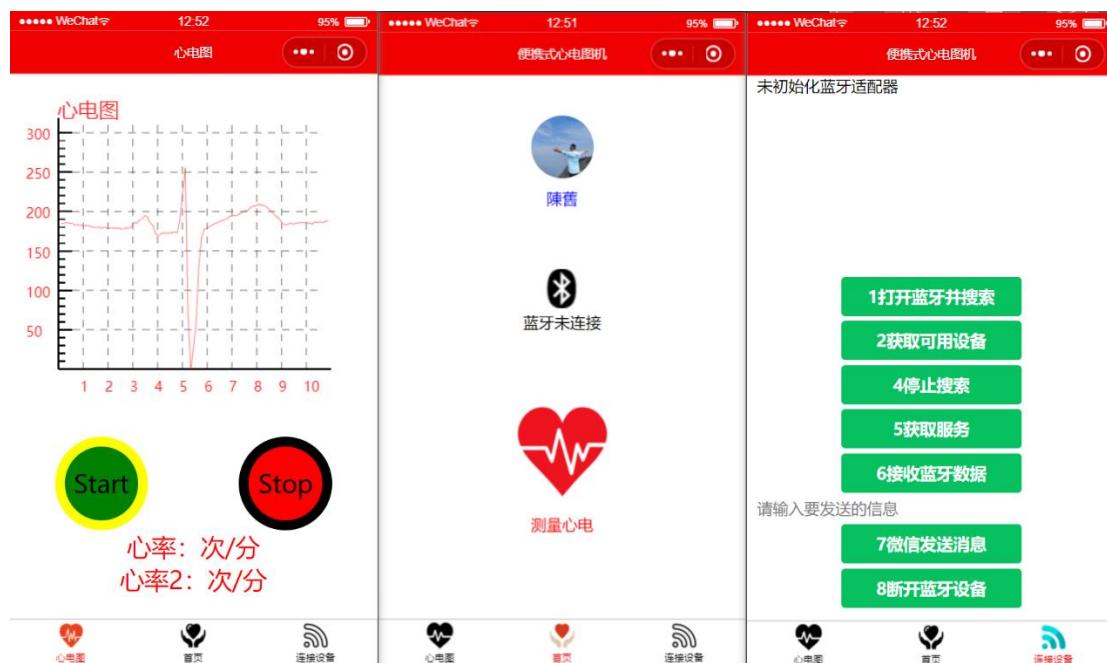


Figure 42. 小程序的三个页面布局

### (2) 蓝牙通信

#### i. 小程序蓝牙介绍[18]

①功耗低：BLE4.0 包含了一个低功耗标准(Bluetooth Low Energy)，可以让蓝牙的功耗显著降低。

②蓝牙终端：我们常说的硬件设备，包括手机，电脑等等。

③UUID：是由字母和数字组成的 40 个字符串的序号，根据硬件设备有关联的唯一 ID。

④设备地址：每个蓝牙设备都有一个设备地址 deviceId，但是安卓和 IOS 差别很大，安卓下设备地址就是 mac 地址，但是 IOS 无法获取 mac 地址，所以设备地址是针对本机范围有效的 UUID，所以这里需要注意。

⑤设备服务列表：每个设备都存在一些服务列表，可以跟不同的设备进行通信，服务有一个 serviceId 来维护，每个服务包含了一组特征值。

⑥服务特征值：包含一个单独的 value 值和 0 ~ n 个用来描述 characteristic 值（value）的 descriptors。一个 characteristics 可以被认为是一种类型的，类似于一个类。

⑦ArrayBuffer：小程序中对蓝牙数据的传递是使用 ArrayBuffer 的二进制类型来的，所以在我们的使用过程中需要进行转码。

## ii. 微信小程序 API 接口

API，即应用程序接口，微信小程序为蓝牙模块提供了 18 个 API 接口，其中低功耗蓝牙 9 个，传统蓝牙 9 个，极大地方便了开发。整理的 API 表格如 Figure 43 所示：

API 名称	说明
openBluetoothAdapter	初始化蓝牙适配器，在此可用判断蓝牙是否可用
closeBluetoothAdapter	关闭蓝牙连接，释放资源
getBluetoothAdapterState	获取蓝牙适配器状态，如果蓝牙未开或不可用，这里可用检测到
onBluetoothAdapterStateChange	蓝牙适配器状态发生变化事件，这里可用监控蓝牙的关闭和打开动作
startBluetoothDevicesDiscovery	开始搜索设备，蓝牙初始化成功后就可以搜索设备
stopBluetoothDevicesDiscovery	当找到目标设备以后需要停止搜索，因为搜索设备是比较消耗资源的操作
getBluetoothDevices	获取已经搜索到的设备列表
onBluetoothDeviceFound	当搜索到一个设备时的事件，在此可用过滤目标设备
getConnectedBluetoothDevices	获取已连接的设备
createBLEConnection	创建BLE连接
closeBLEConnection	关闭BLE连接
getBLEDeviceServices	获取设备的服务列表，每个蓝牙设备都有一些服务
getBLEDeviceCharacteristics	获取蓝牙设备某个服务的特征值列表
readBLECharacteristicValue	读取低功耗蓝牙设备的特征值的二进制数据值
writeBLECharacteristicValue	向蓝牙设备写入数据
notifyBLECharacteristicValueChange	开启蓝牙设备 notify 提醒功能，只有开启这个功能才能接受到蓝牙推送的数据
onBLEConnectionStateChange	监听蓝牙设备错误事件，包括异常断开等等
onBLECharacteristicValueChange	监听蓝牙推送的数据，也就是 notify 数据

Figure 43. 蓝牙 API

### iii. 蓝牙通信的正常流程

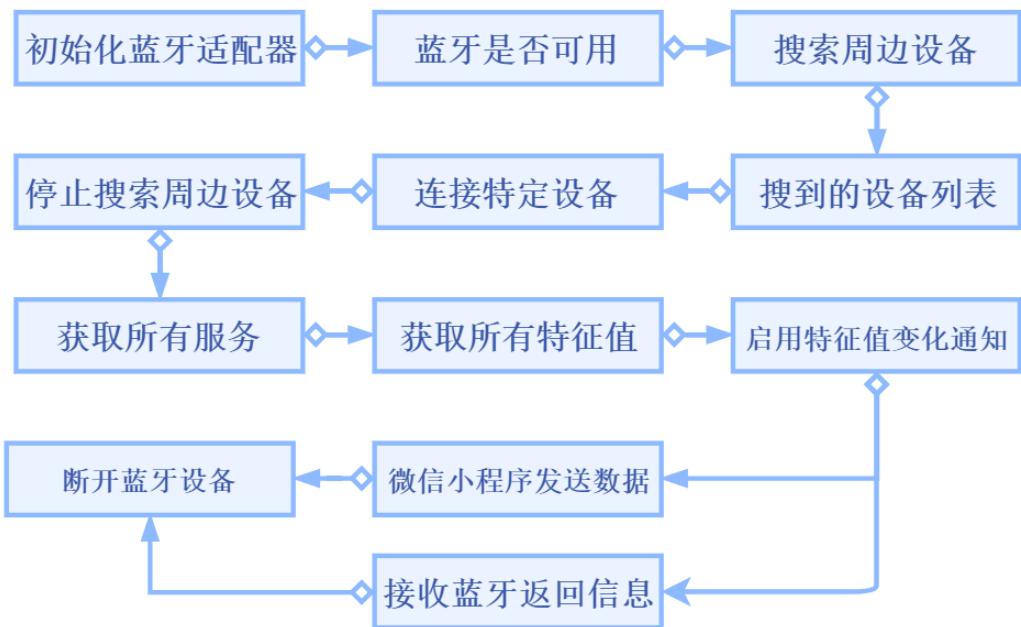


Figure 44. 蓝牙通信流程图

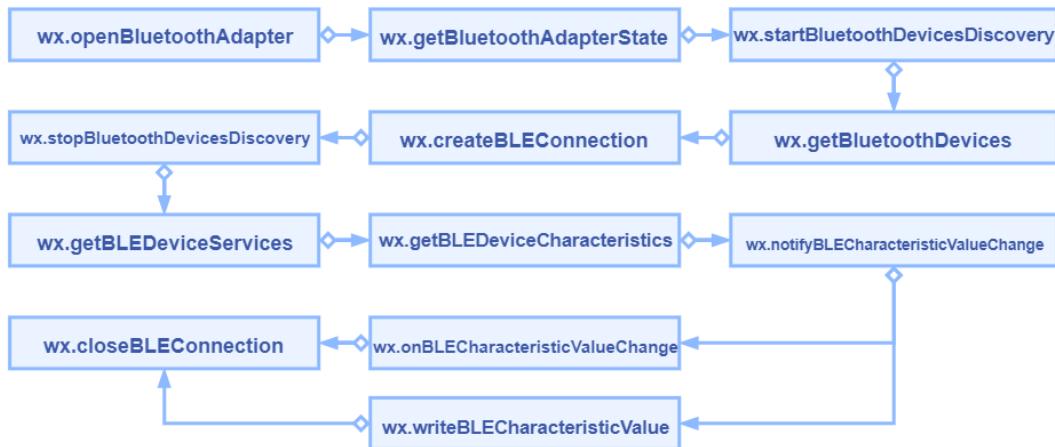


Figure 45. 蓝牙通信流程图对应 API

iv. 蓝牙传递数据说明

对如何传递数据进行如下说明：服务端定好一个参数，客户端可以对这个参数进行读、写、通知等操作，即为特征值；参数一个不够用，所以有多个特征值，并进行分类，分出的类即为服务；所以就是一个设备有多个服务，每个服务有多个特征值，每个特征值有它的属性；而小程序的蓝牙通信必须要有读、写、通知三个属性，也就是选用的特征值对象必须包含 read、write 和 notify 功能。53[19]

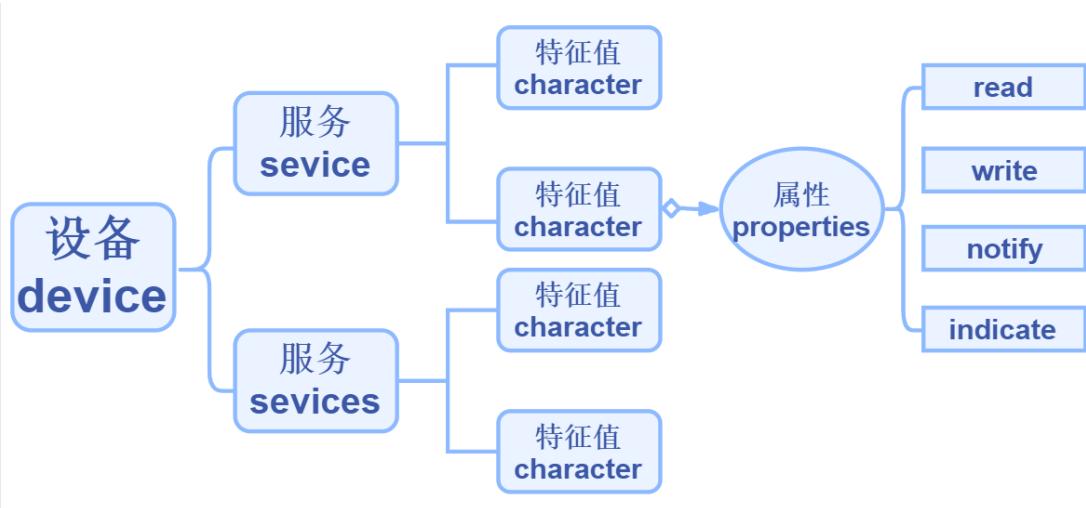


Figure 46. 蓝牙通信中设备、服务与特征值的关系

#### v. 具体编写代码与调试过程

①借鉴网上的例程[18]实现整体框架，然后进行调试与修改。因为手头没有 Ego1 开发板，小程序仅支持低功耗蓝牙 4.0，无法连接手机和电脑的蓝牙，不知如何测试。后来发现好多例程中都提到使用 HC 系列的蓝牙模块，于是购买 HC-08 蓝牙模块进行调试；之后是对代码的修改和调试。下面只摘取蓝牙代码中的关键部分进行解释说明[20]。

#### ②搜索设备

注意例程中的 services 可能是写死的，所以可能出现搜不到设备或者搜到含这个 UUID 的设备，建议开发过程中注释掉这部分；确定自己设备服务后也可以写死，搜索出指定的设备。

```

74 //搜索设备，此处可以设置搜索特定设备
75 lanyatest3(event) {
76     var that = this;
77     wx.startBluetoothDevicesDiscovery({
78         //services: ['FFF0'], //如果填写了此UUID，那么只会搜索出含有这个
79         //UUID的设备，建议一开始先不填写或者注释掉这一句
80         success: function (res) {
81             that.setData({
82                 info: "搜索设备" + JSON.stringify(res),
83             })
84             console.log('搜索设备返回' + JSON.stringify(res))
85         }
86     },

```

Figure 47. 搜索设备函数

### ③获取蓝牙特征值并启用特征值变化 notify

读取到的 UUID:

第1个UUID:0000180A-0000-1000-8000-00805F9B34FB	bluetooth.js? [sm]:153
第2个UUID:0000FFF0-0000-1000-8000-00805F9B34FB	bluetooth.js? [sm]:153
第3个UUID:0000FFE0-0000-1000-8000-00805F9B34FB	bluetooth.js? [sm]:153
第4个UUID:00001800-0000-1000-8000-00805F9B34FB	bluetooth.js? [sm]:153
第5个UUID:00001801-0000-1000-8000-00805F9B34FB	bluetooth.js? [sm]:153

Figure 48. 读取到连接设备的 UUID

这里需要注意，设备会搜索到多个服务，而不同设备虽然可能有一样的服务，但在服务列表中的顺序不一样，比如我们用到的 UUID，在 HC-08 在服务列表数组中对应下标为 2 的元素，而 Ego1 上蓝牙模块则对应下标为 1 的元素。因为两者用到的 UUID 是一样的，可以直接写死在这里。

```
162 //获取连接设备的所有特征值
163 lanyatest8(event) {
164     var that = this;
165     var myUUID = "0000FFE0-0000-1000-8000-00805F9B34FB"; HC-08和Ego1的蓝牙一样，可以写死在这里
166     //var myUUID = that.data.services[2].uuid;//对应HC-08蓝牙模块的，第3个
167     //var notifyServiceId = that.data.services[1].uuid; //对应Ego1上蓝牙模块的，第2个
168     //var myUUID = that.data.servicesUUID;//具有写、通知属性的服务uuid
```

Figure 49. 在获取所有特征值函数中将 myUUID（即用到的服务）写死

函数中将 myUUID（即用到的 UUID）写死，之后调用获取所有特征值函数 wx.getBLEDeviceCharacteristics，通过 for 循环找出指定服务中具有读写和通知属性的特征值，记为 myUUID。

```
getBLEDeviceCharacteristics
特征值: 0000FFE1-0000-1000-8000-00805F9B34FB
notifyServiceId: 0000FFE0-0000-1000-8000-00805F9B34FB
notifyCharacteristicsId: 0000FFE1-0000-1000-8000-00805F9B34FB
writeServiceId: 0000FFE0-0000-1000-8000-00805F9B34FB
writeCharacteristicsId: 0000FFE1-0000-1000-8000-00805F9B34FB
readServiceId: 0000FFE0-0000-1000-8000-00805F9B34FB
readCharacteristicsId: 0000FFE1-0000-1000-8000-00805F9B34FB
device getBLEDeviceCharacteristics: ▶ [{...}]
启用notify的服务Id 0000FFE0-0000-1000-8000-00805F9B34FB
启用notify的notifyCharacteristicsId 0000FFE1-0000-1000-8000-00805F9B34FB
```

Figure 50. 获取到的具有读、写、notify 功能的特征值

```
characteristicId: 0000FFE1-0000-1000-8000-00805F9B34FB  
serviceId:0000FFE0-0000-1000-8000-00805F9B34FB
```

Figure 51. 具体使用的可用于蓝牙通信的服务和特征值

然后是启用特征值变化 notify，开启 notify 功能才能进行读写操作，所以后面在 wx.notifyBLECharacteristicValueChange 这个 API 中的 serviceId 需要设置为前面找出的那个特征值 myUUID，这样就可以监听特征值变化，可以接收蓝牙模块发送过来的数据。

#### ④接收蓝牙返回信息

```
wx.onBLECharacteristicValueChange(function (res) {  
  
    console.log("characteristicId: " + res.characteristicId)  
    console.log("serviceId:" + res.serviceId)  
    console.log("deviceId" + res.deviceId)  
    console.log("Length:" + res.value.byteLength)  
    console.log("value:" + res.value)  
    var b = new Uint8Array(res.value); //解析为10进制数据  
    console.log("转化后的十进制数组: " + b);  
    for (var i = 0; i < b.length; ++i) {  
        //console.log('b:' + b[i]);  
        app.globalData.buffer.push(b[i]);  
        //console.log('app.globalData.buffer:' + app.globalData.buffer);  
    };  
});
```

Figure 52. 接收蓝牙返回信息函数

开启特征值变化 notify 之后，就可以接收蓝牙返回的数据，注意得到的数据是 ArrayBuffer 类型的。可以通过 ab2hex 函数转化为十六进制数据，用于检验收发是否一致。

可以用 Uint8 对二进制数组解析为十进制，直接定义一个新的 Uint8Array 数组，就转化为 10 进制数据。最后通过 for 循环将每个包（蓝牙模块大于 20 字节的数据是分包发送的，每个包 20 字节的数据）中的数据一个一个 push 进全局数组变量 buffer 中，然后就可以转到心电图界面进行动态显示了。

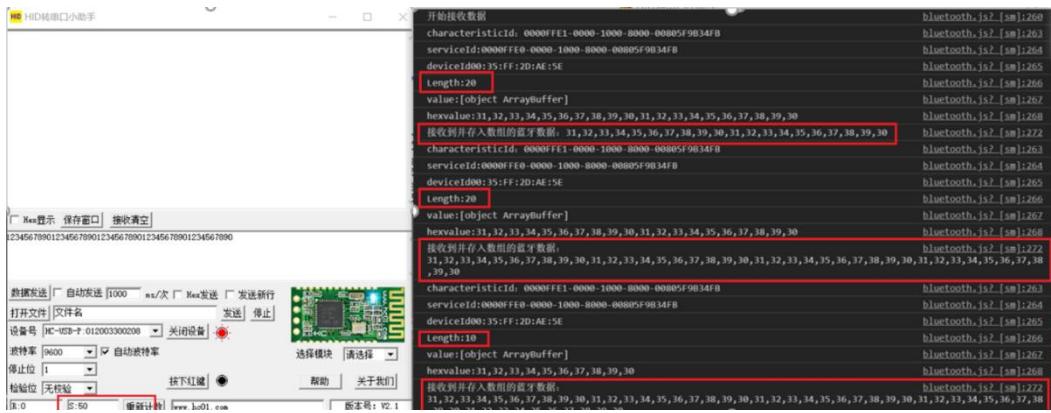


Figure 53. 分包接收调试结果

可以看出串口助手发送了 50 个字节的数据，然后在串口助手依次收到 20、20、10 个字节的三个包的数据，然后依次存入接受数组 buffer 中。

#### ⑤发送数据

出现的一个问题是小程序发送数据报错（发送数据会报错，后发现不是代码的问题，是输入的数据有问题，注意输入数据不能是中文，改为输入数字就可以发送数据，在串口助手显示蓝牙模块接收到数据）。

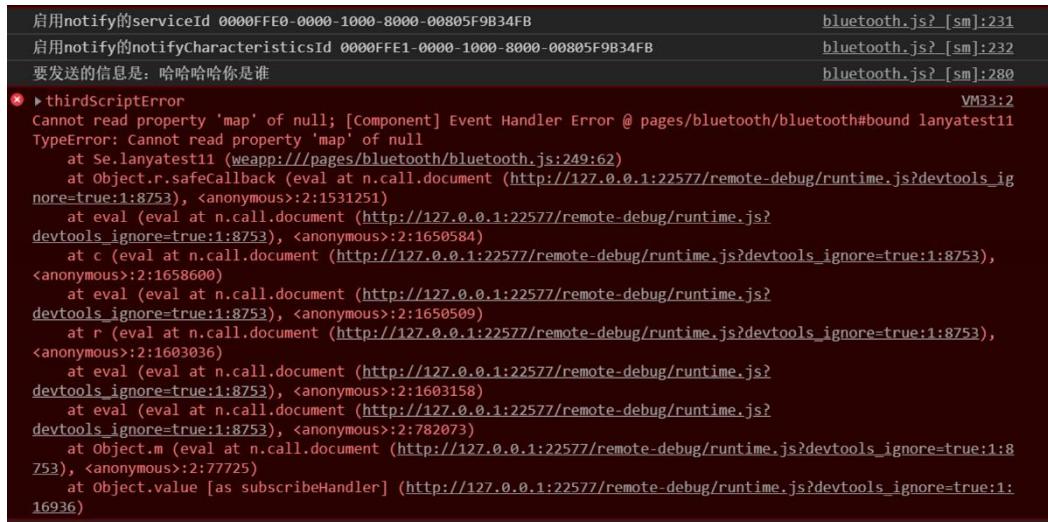


Figure 54. 发送中文报错问题

#### (3) 动态显示

动态显示模块的功能是将蓝牙通信模块接收到的心电数据按照一定的刷新频率和显示范围绘制出坐标图，以便直观地将前面所有模块得到的结果进行展示。

另外由于实际的需要，还需要对当前显示进行开始和暂停，然后将心率和心电分析的实时结果实时展示在下方。

### ①绘图模块

绘图模块使用了 canvas 模块，canvas 模块是 JavaScript 支持的绘图模块，可以在页面中划定一块长度和宽度自定的区域，然后在此区域内进行绘制，首先绘制坐标图和网格点，然后将接受到的蓝牙全局数组变量按照固定的长度切片，然后将切片数组显示在坐标图内。

#### a. 表格初始化

绘制图片需要调用 initChart 初始化函数来对坐标图进行初始化创建，初始化的第一步首先创建 canvas 绘图上下文 CanvasContext 对象，然后使用 beginPath 函数创建一个路径，当需要调用 fill 或 stroke 函数时会使用路径进行填充和描边，setStrokeStyle 和 setFillStyle 将填充和描边的颜色设置为黑色。setLineWidth 将线条宽度设置为 1 个像素。然后调用屏幕自适应函数 getEleWidth 定义坐标图左上角在屏幕上的相对横纵坐标。同样地，使用屏幕自适应函数 getEleWidth 分别定义坐标图的右下角和在 x 轴 y 轴方向上的宽度和高度。进行波形绘制之前需要调用 clearRect 函数，清空左上角和右下角之间的矩形区域，这是因为第二次绘制时如果不清空之前绘制的波形，坐标图上将会同时显示两次绘制的波形，看上去有“重影”。然后开始绘制坐标轴，使用 moveTo 函数将路径移动到之前定义好的坐标轴左上(leftTopX, leftTopY)，但并不创建线条，然后使用 lineTo 函数增加一个新点坐标轴左下(leftBottomX, leftBottomY)，然后创建一条从上次指定点到目标点的线。再用 stroke 方法来画出线条。继续使用 lineTo 函数画出到右下角(rightBottomX, rightBottomY)的线条，然后设置字体（即图片标题“心电图”）的大小，以及标题的文本内容，标题的起止坐标。然后调用 drawYscale 和 drawXscale 函数绘制出坐标轴上的尺度划分，使用 drawCharts 函数绘制出当前蓝牙全局数组的切片数组对应的波形。然后 stroke 函数画出当前路径的边框，默认颜色为黑色，最后 draw 函数将之前在绘图上下文中的描述（路径、变形、样式）画到 canvas 画布中。

#### b. 绘制坐标轴和刻度

initChart 初始化表格函数中调用了绘出 Y 轴以及尺度划分的函数 drawYscale，其内部主要完成了轴的绘制和刻度的绘制，其中刻度分为大刻度和小刻度，大刻度的长度比小刻度更长，且标注了当前刻度对应的数字，大刻度之间的间隔是 50，小刻度之间的间隔是 10，两个相邻的大刻度之间有 4 个小刻度。调用 drawDashLine 函数绘出虚线网格图。

X 轴的函数 drawXscale 也与之类似，但是不需要画出刻度和标注数字，因为 X 轴的含义是时间。

#### c. 绘制网格虚线

drawDashLine 函数是 drawXscale 和 drawYscale 中调用的绘制网格虚线函数，drawDashLine 中定义了虚线的宽度和径向间隔和法向间隔。X 轴上与 Y 轴上固定间隔的相互垂直的虚线构成了网格线。

#### d. 绘制波形曲线

init 初始化函数中调用 drawCharts 来绘制心电信号的波形图，oneScaleX 意思是将 x 轴分为需要显示的切片数组的长度的份数，即波形对应的点的横坐标可以用离原点的格子数加上原点离屏幕边缘的距离表示，纵坐标同理。然后使用 moveTo 和 lineTo 函数将波形点连起来。

#### e. 获取屏幕自适应宽度

以宽度 480px 作为宽度的自适应。

具体代码参照附件中代码。

### ②显示模块

#### a. 方案一

通过绘图模块得到了一系列时间上连续的波形图，方案一的想法是将绘制得到的波形图以 jpg 格式保存到本地，然后使用 swiper 模块按照一定的帧率滑动 swiper 模块，形成动态的效果。

swiper 是微信小程序自带的模块，使用起来非常方便，只需要改变传入参数就可以修改滑动模块的属性，按住滑动模块即可暂停，松开滑动模块即可继续播放。indicator-dots 是否显示滑窗上的指示点，默认 false，indicator-color:指示点颜色，indicator-active-color:选中的指示点颜色，autoplay: 是否自动切换，默认: false。interval: 自动切换时间间隔。duration: 滑动动画时。vertical 是否改成纵向，默认 false。图中采取了 50ms 的切换间隔，也就是 20fps 的显示效果。

方案一存在缺点：保存到本地的图片会占用大量磁盘空间，而且运行程序后也不会删除图片，虽然显示的帧率高却会造成用户手机的负担，另外，如果用户后台运行小程序时其他程序也向手机中保存图片，会显示不相关的内容。因此采用了第二种方案。

#### b. 方案二

通过调用时间函数，将 canvas 绘制好的图片显示在页面上，然后按照固定的时间间隔，改变全局蓝牙数组的切片位置，并清空前一次绘制的波形曲线，重新绘制波形，并实时显示在页面上。

开始函数如 Figure 55 所示

```
onStartHandler() {
    if (!interval) {
        interval = setInterval(() => {
            this.setData({
                time: this.data.time + 1,
                displayTime: this.parseTime(this.data.time)
            })
        }, 10);
    }
},
```

Figure 55. 开始函数

点击界面上的 Start 按钮之后触发该函数，会将当前显示的数组根据时间如同移位寄存器一样改变在蓝牙全局数组中的切片部分，并将时间轴上的间隔数设置为 10.

停止函数的原理类似，会使当前显示的数组保持不变。

页面加载函数，每次调用都会运行一次表格初始化函数 initChart()

```
onLoad: function (options) {
    this.setData({
        heartRate2: app.globalData.heartRate
    });
    this.initChart()
    console.log('onLoad')
    console.log(arr)
},
```

Figure 56. 页面加载函数

parseTime 是根据时间改变显示数组的函数，通过改变 isInteger 判断函数内的时间相关表达式来改变图片刷新的频率和波形每次移动的距离，通过 slice 函数来切片全局蓝牙数组并控制切片数组的长度，然后将得到的切片数组返回，调用 onload 函数重新加载图表。此时 init 初始化函数中的 clearRect() 就会起到清空前一次波形的曲线的作用，从而产生波形移动的效果。而 Stop 按钮调用的 stopTime 函数的功能也类似，区别是会 slice 切下的数组内容不会发生改变。

```

parseTime() {
  if (Number.isInteger(this.data.time % 6000/100)) {
    var ss = parseInt(this.data.time % 6000 / 3);
    if (ss < arr1.length)
    {
      arr = arr1.slice(ss, ss + 300);
      console.log(arr);
    }
    else {
      arr = [0];
    }
    this.data.list = arr
    this.onLoad()
  }
  return `${arr}`;
},

```

Figure 57. parseTime 函数

最后在 wxss 和 wxml 中设置页面的样式和布局，原则是美观和简洁。

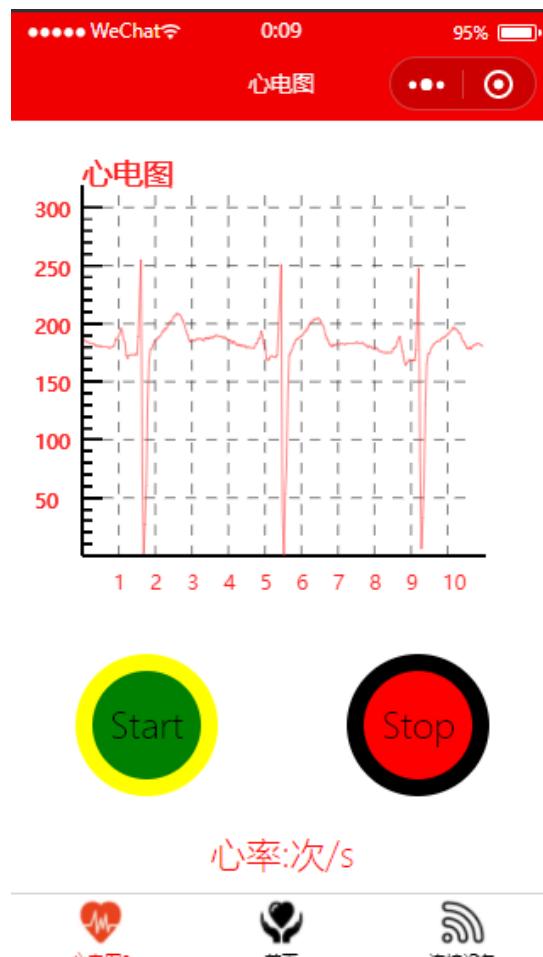


Figure 58. 动态显示效果图

#### (4) 小程序整合

主要是实现不同界面的数据传递，将原来对固定数组的动态显示改为对蓝牙接收到的数据存入一个全局数组变量并动态显示。

整合过程关键有两点，一个是将接收到的二进制数据转化为10进制数据并存到一个数组中；另一个是全局变量的数据传递。

#### (5) 心率计算

①峰值检测算法：选取合适的长度length对存储接收数据的数组buffer截取，得到n个数组arr2；通过getMaxIndex函数[1]获取每个数组的最大值下标，这些最大值下标组成一个新的数组arr3；然后计算相邻最大值间隔的数据个数，即用for循环计算：  
 $length - arr[j] + arr[j+1]$ ，进而得到数组arr4，取均值得到平均间隔数据个数HR。设采样率为sampling\_rate，即前端电路1s采集到sampling\_rate个心电数据，然后心率可以通过 $sampling\_rate * 60 / HR$ 计算得到。该算法需要选取合适的截取长度和取样数组个数，否则会出现某一段没有实际峰值或者有两个实际峰值的情况。[21]

②上升沿阈值检测：对整个存储接收数据的数组buffer进行阈值检测，规则是设定一个阈值x，当 $buffer[i] \leq x$ 且 $buffer[i+1] \geq x$ 的时候，记录下标i，放进新的数组arr中。然后对arr中的数组取均值得到平均间隔数据个数。之后的心率计算方法与峰值检测一致。该算法必须要在接收到的心电数据具有明显的上升沿或者下降沿才可以保证精确度。根据I导联心电波形的特点，可以看出有比较明显的上升沿，只要选取合适的阈值，可以保证检测的数据准确性。

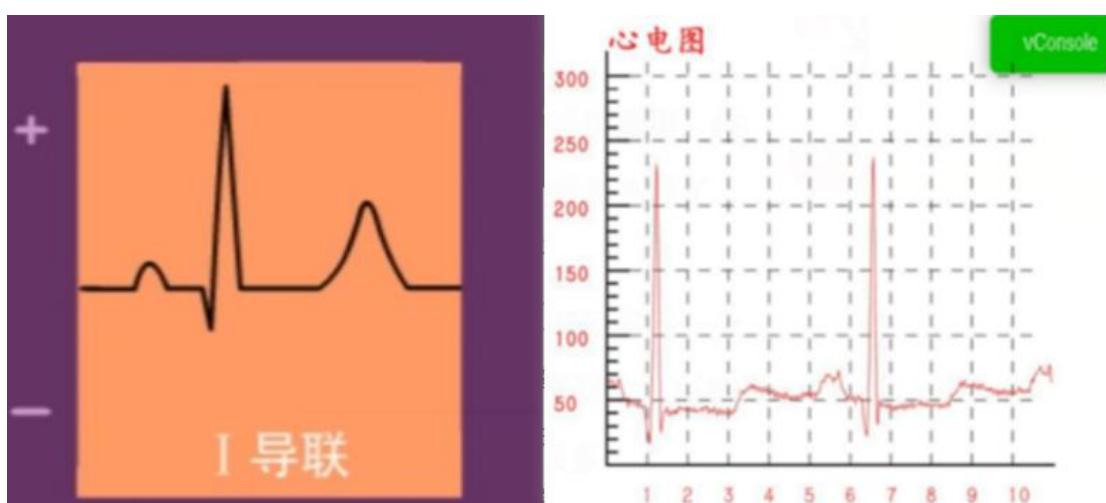


Figure 59. I 导联心电波形（左）和测试心率计算功能用心电图波形（右）

### ③两种心率算法的测试

```
峰值心率间隔数组: 105,103,100,99,103,104,103,101,102,103,103  
最高心率的平均间隔数据个数: 102.36363636363636  
心率: 59次/分 (算法1)  
阈值点位置: 75,180,283,383,482,585,689,792,893,995,1098,1200,1201  
阈值心率间隔数组105,103,100,99,103,104,103,101,102,103,102  
阈值心率的平均间隔数据个数: 102.27272727272727  
心率: 59次/分 (算法2)
```

Figure 60. 两种心率算法对比

可以看出峰值检测选取合适的参数和阈值检测得到的数据得到的结果几乎一样，根据实际情况可以选择一种合适的算法来计算心率。而且在尝试第二种算法的时候，也找出了第一种算法的错误，并进行了修正。

### (6) 小程序调试

采用 HC-08 蓝牙模块、HC 专用测试架和 HC-USB-P 串口调试助手（即 HID 转串口小助手）进行调试；模拟心电数据通过 matlab 进行了预处理，首先归一化到 0-255 之间，然后取整，转化为 16 进制数据方便串口助手 Hex 发送；小程序连接蓝牙并接收数据，转化为 10 进制后绘制心电图，也显示心率计算结果。



Figure 61. HID 转串口小助手界面和 HC-08 蓝牙模块

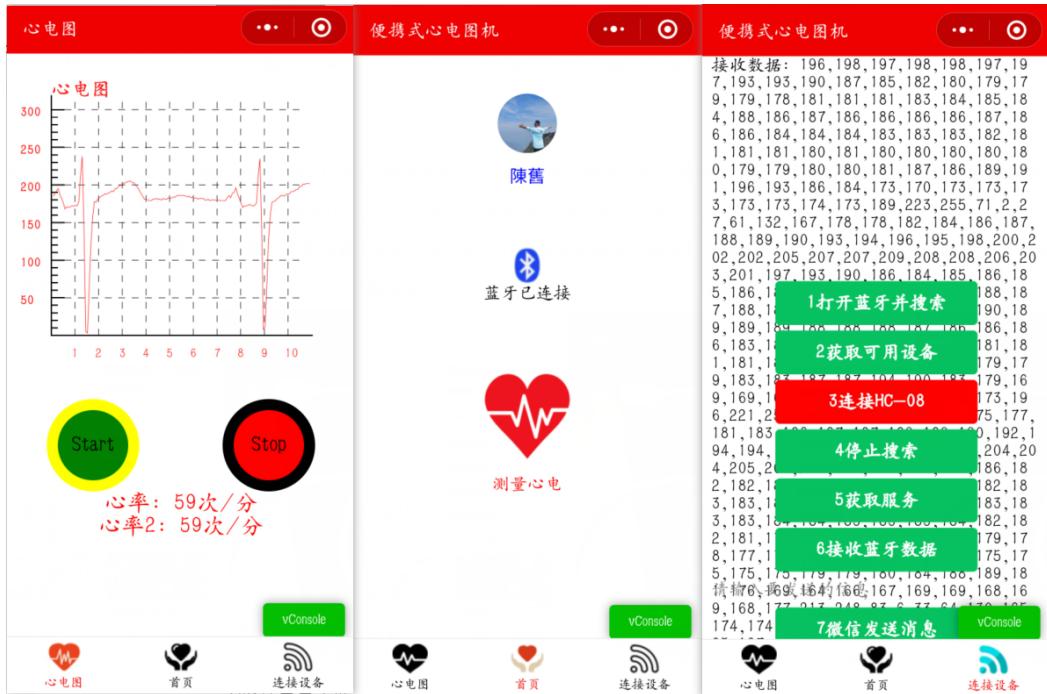


Figure 62. 调试中的三个界面

具体调试过程：

- HC-08 蓝牙模块先通过 HC 专用测试架接到电脑 USB 口；
- 打开 HID 转串口助手，选择 HC-08 模块，打开设备；
- 在小程序连接设备界面完成设备连接。依次点击按钮到“6 接收蓝牙数据”，注意“2 获取可用设备”点击后可能反应比较慢，多点几次，等待出现“3 连接 HC-08”按钮并点击；“5 获取服务”点两次，待上方控制台显示“开启 notify 成功，准备接收数据”；最后点击“接收蓝牙数据”即可。
- 模拟心电数据通过 matlab 进行了预处理，首先归一化到 0-255 之间，然后取整，转化为 16 进制数据
- HID 转串口助手选择使用 Hex 发送模式，发送上面得到的 16 进制数据；
- 在连接设备界面显示接收并转化得到的 10 进制数据
- 转到主页面可以看到显示蓝牙已连接
- 转到心电图界面点击 start 可以动态显示模拟心电图，stop 可以暂停，在下方显示两种心率算法计算得到的心率值。

具体调试过程请看附件中的视频“蓝牙模块 HC-08 与小程序通信演示视频（模拟心电数据）”。

## 五、人员分工

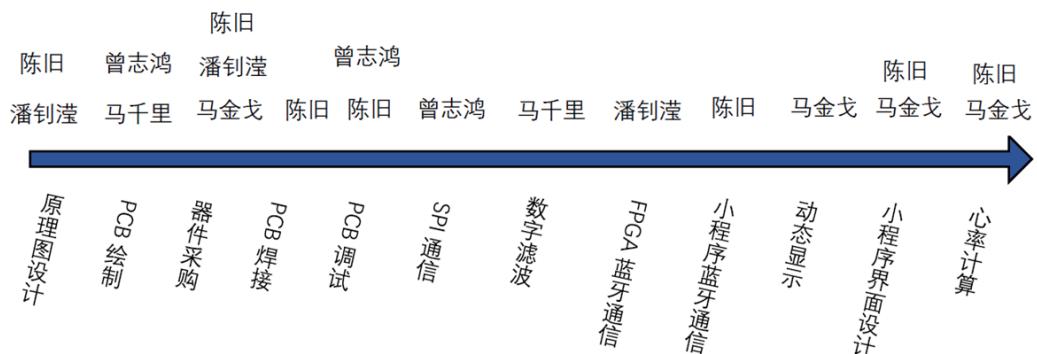
潘钊滢：原理图设计与绘制，器件与仪器采购、数字滤波中陷波器仿真、FPGA 到小程序的蓝牙通信

马金戈：器件与仪器采购，小程序开发的页面设计、动态显示和心率计算

曾志鸿：PCB 绘制、SPI 通信、PCB 调试

马千里：PCB 绘制、数字滤波

陈旧：原理图设计与绘制，PCB 板焊接，小程序开发的页面设计、蓝牙通信、心率计算和整合



## 六、作品展示操作说明

最终成果实现了 Matlab 滤波算法检验和 FPGA 到小程序的蓝牙通信，证明了滤波算法的有效性、蓝牙模块和小程序模块的可用性。此外，SPI 通信部分也通过了仿真测试。

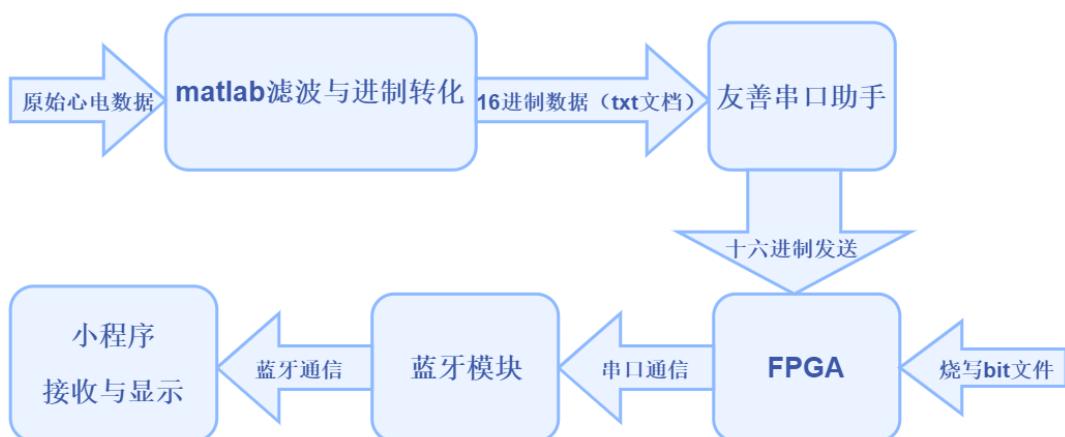


Figure 63. 成果展示流程图

1. 通过 matlab(NoiseAndCarrier.m)对 MIT-BIH 心律失常数据库中的数据(data\_in.txt, 即数据库中的 100.hea) 进行滤波处理, 得到一个 16 进制文档 data.txt
2. FPGA 烧写 “top.bit” 文件
3. 然后通过电脑上的友善串口助手, 16 进制发送上述得到的文档 data.txt 到 FPGA
4. FPGA 内部通过串口通信将数据传给蓝牙模块, 然后与微信小程序进行蓝牙通信。
5. 在小程序连接设备界面完成设备连接, FPGA 上蓝牙模块的灯不闪烁, 表明设备连接成功。 (具体连接步骤见小程序调试部分)
6. 点击连接设备界面的“6 开始接收数据”按钮, 在连接设备界面可以看到接收到的 10 进制数据, 可与发送前对比
7. 数据接收完毕后, 转到心电图界面, 点击 start, 可以看到心电图的动态绘制, 下方是两种心率算法得到的结果。此外点击 stop 可以暂停绘制。

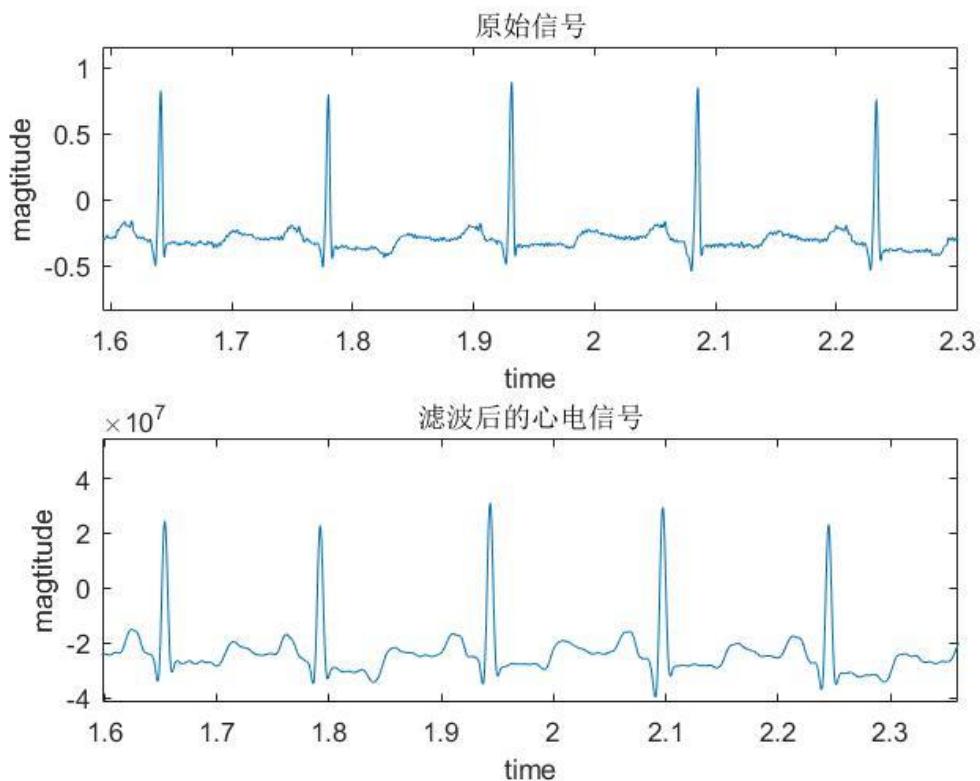


Figure 64. 滤波前后的心电信号

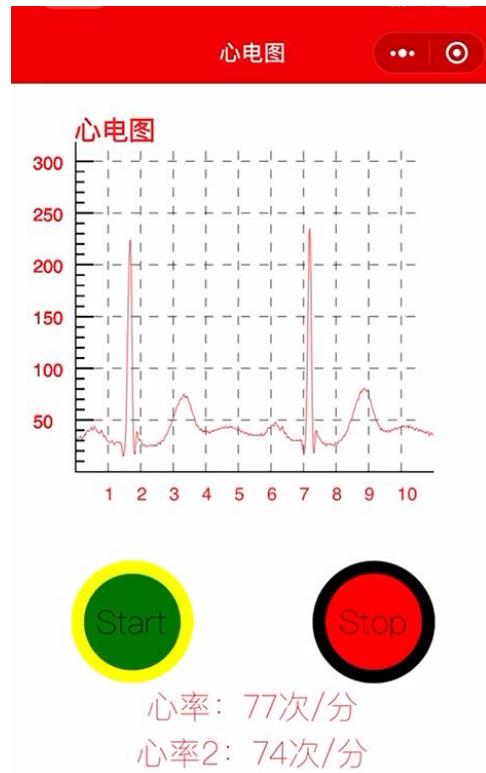


Figure 65. 小程序端显示的心电图

可以看出绘制出的心电数据（即滤波后的心电数据）和 matlab 显示的波形是大体一致的。但因为归一化后精度下降，导致绘制出来的图仍是有毛刺的。

心率计算的结果差别不大。在视频中可以看出心率算法 1（即峰值检测算法）的结果是不变的，因为只决定于前一段数据；而心率算法 2（即上升沿阈值检测算法）的结果随着接收数据的增多在不断地改变，最终趋于稳定。所以心率算法 1 的数据准确性较差，且前面提到普适性差，需要修改参数；而心率算法 2 准确性好，普适性也好，更适合作为心率算法使用。

具体调试可以参见附件视频“FPGA 与小程序蓝牙通信演示视频（心电数据库某异常心电数据）”。此外，我们的小程序并未公开发布，测试时联系负责开发小程序的同学，添加为开发者后才能扫二维码进入小程序。

注：

①因为之前小程序调试用到的模拟心电数据与 FPGA 用到的心电数据库数据的采样率和波形特点都有差别，所以需要修改代码进行适配。主要是修改动态绘图一次绘制的数组长度，图像移动速度，数据采样率，心率算法 1 的截取数组长度和截取数组个数，心率算法 2 的阈值。

②发送的 txt 文档需要有特定的格式，每个数据为两位 16 进制数据，数据之间空格隔开，不要有换行。

③看小程序连接设备界上方会显示当前状态信息，接收到数据后可以与与发送数据对比，检验数据是否发送正确。

④手机性能会影响接收和显示的效果。发送的数据过多，手机性能不好的话会卡住，反应好久才能接收完毕并显示；动态绘图的刷新速度也会受手机性能影响，调整参数增快刷新速度，对性能差的手机无法显示出来。（华为荣耀 8X 接受数据很慢；华为荣耀 8X 和小米 K20 两个手机刷新速度有限制；iPhone11 无论是接收数据还是刷新，速度都很快。）

⑤两种心率算法对比：总体来说上升沿阈值检测算法普适性好，更加准确；而峰值检测需要针对不同的心率修改参数，普适性很差。发现峰值检测算法针对正常人心率范围（60–100 次/分）普适性较差，例如针对 60 次/分和 100 次/分，采用相同的数组截取长度总有一个心率计算会出现很大的偏差，因为会出现一个截取数组长度内存在两个峰值或者没有峰值的情况；而上升沿阈值检测算法则比较理想，因为发现心电数据波形上具有明显而且陡峭的上升沿，所以只要选择合适的阈值，不会出现漏掉某个周期或者一个周期内采样多次的情况，普适性较好。

⑥归一化会使得数据精度下降，实际小程序绘制的滤波后的心电信号仍有毛刺

## 七、经验总结

1. 设计原理图经验总结：绘制原理图芯片选型和外围电路设计是个很麻烦的事情，需要仔细阅读芯片手册，确定合适的外围电路元器件值。我们小组第一次阅读 datasheet 是 ADS1292R 这样一个高度集成的复杂芯片，阅读时不容易抓到重点，后续阅读供电模块的简单芯片时发现 datasheet 一般有比较完善的参考电路等。重读 ADS1292R 的 datasheet，我认为需要从 pin configuration 入手，寻找有没有可供参考的电路设计。如果 datasheet 没有，可以寻找该芯片有没有 demonstration board，其设计图也可作为参考电路。设计电路时也需要考虑各个模块的级联，输入输出电流要注意。在我们的设计中，芯

片是否方便购买也没有提前考虑，导致选好芯片并设计好电路后，发现有些芯片买不到，花了很多功夫寻找替代；所以在选芯片的时候一定要看看立创商城有没有，能不能买到，选择销量高一点的比较靠谱，不能只为了设计参考电路方便而随便选一个芯片。其他元器件的选择也要注意，最好都可以在一个商城买到。另外器件的封装也是特别需要注意的地方，阅读 datasheet 时没有注意到集成芯片有两种封装，导致后面元器件不好购买和焊接极其困难的问题。另外电极片开始也是一点思路没有，不知道如何与采集电路连接，后来与老师沟通后确定用裸露的焊盘实现。另外各种接口的设计也是需要全局考虑的，接口分 male 和 female 可能引起混淆，应多加注意。

2. PCB 板焊接的问题：先买了几个 PCB 焊接练习板练了练手，在 b 站对一些焊接注意事项和技巧进行学习了解，对于一般的元器件焊接差不多。因为选了两个 QFN 封装，没有伸出来的引脚，不好焊接，也可能是焊接工具的问题。助焊剂的作用不知道发挥好没，因为用完之后用摄像机看引脚时，会有反光而且不好对焦，看不清是否短路，所以废了很多时间。焊接 QFN 出现了很多引脚短路问题，老师建议多加助焊剂，利用焊锡的重力斜着焊，虽然费了很多工夫，焊废了好几个芯片，最后总算效果还行。然后是备用电路与充放电电路并联的问题，焊接完成整个电路后发现充放电电路没法正常工作，直接供电的指示灯也是一闪一闪的，但是用另一块板只焊接直接供电部分可以正常工作，而且测量到的电压是期望的值，可能是两种供电电路并联带来的 bug。于是充放电电路没有焊接，只用了备用的直接供电电路。另外上述焊接完成测试出问题导致不知所措的情况，带来了不少教训和经验：需要注意 PCB 的焊接顺序，逐级焊接并逐级测试，防止焊完整个电路测试不对无从下手；上电前先测试电路的关键部分，万用表量一下电阻之类的，防止直接上电出问题烧坏电路；最好加入 0 欧姆电阻作隔断前后级电路，方便逐级测试。此外还有焊接 micro-usb 堵住接口、焊盘被焊掉的情况，这些也是很糟糕，因为有些失误是没法补救的，导致前功尽弃，只能重新再来。

3. 焊接技巧总结：拖焊、清理多余焊锡的时候轻轻从引脚上滑过，利用惯性带锡，不要太使劲压着引脚。一般是先焊上一侧引脚固定住再焊接，焊点形状带球带刺的情况说明没焊好，可以加点助焊剂，最好是裙带状。电烙铁长时间不用一直在加热对焊头不好，氧化后不好用，最好加一层锡保护。焊头一般用刀头就行，温度设置到 330–350°C。焊这种贴片类型的话，锡线用细一点的好（0.6mm 可以）。焊的时间不要太长，助焊剂会挥发，引脚会氧化。处理引脚短路问题，多用助焊剂，可以用干净的烙铁头直接沾走，也可以用吸锡带放到短路的位置，烙铁头压一下，注意不要使力气。焊锡不要往烙铁头上送，往焊盘和引脚上送。焊接过程中会有很多烟气，可以准备个小风扇，注意通风。

4. 小程序学习：主要是在 b 站搜索教程，先是通过四个简单的实战小程序进行初步了解；然后学习了黑马程序员的教程，对小程序的基础功能和框架有了了解，并选用 canvas 实现绘图功能，完成了基本框架的设计；最后通过“清华大学+学做小程序”，这个教程对框架和功能的讲解很清晰，对小程序的编写有了更深入的认识。然后简化界面，将不同功能放到不同页面实现，方便整合。分工实现蓝牙通信和动态显示两部分功能。之后是针对要实现的功能分别进行查阅资料，编写代码。

5. 小程序开发经验总结：手机电脑的蓝牙没法用作调试，需要购买一个支持低功耗蓝牙 4.0 蓝牙模块（本设计中用到 HC-08）用作测试；搜索设备的时候可能需要等待一会才会显示出搜索到设备；另外网上的示例代码有的是把设备写死了，可能与自己用的蓝牙模块不一致就搜不到设备；需要查阅蓝牙模块手册确定哪个服务具有可以读、写和通知功能的特征值，然后后面调用这个特征值才可以顺利进行蓝牙通信；在对接收到的 ArrayBuffer 数组进行处理的时候，没有认识到它的本质，然后尝试了好多进制转换和数组拼接方法，都没法把分包接收的数据转化为十进制并存入一个数组，后来查阅资料才知道 ArrayBuffer（二进制数组）并不是真正的数组，而是类似数组的对象，之前一直按照数组的格式对其进行处理，出现了很多问题。绘制的时候出现重影，原因是没有在下一次绘制之前清除画布。在动态绘图调试的时候，发现在电脑上可以显示动态绘图，但是到手机上就显示不了，而另一个同学的手机行；还有种情况是用真机调试功能不行，用预览功能行；动态显示调试的时候选用的数组是周期的，所以看不出来会动，导致不成功，用于测试的数据也需要考虑好，不能随便设置。push 是很方便的一种拼接数组的方法。对于数据的位数之前没有考虑到，在后面才意识到 ADS1292R 的 ADC 是 24 位的，而模拟心电数据发过来是一个字节（也就是 8 位）代表一个数据，接口关系上可能存在问题，后来和做 FPGA 的同学沟通后在前面做一下归一化处理后再发送。

6. 数字滤波设计经验总结：数字滤波设计中我们首先使用了 matlab 仿真，涉及到输入信号的使用，我们项目中使用了 MIT-BIH 数据库，但该数据库使用了特殊的文件格式，一时之间无处下手。后来在数据库官网找到了官方给出的读取数据程序 rddata.m，把数据库的文件读到 matlab 的工作区；其次仿真信号时，查找手册发现 ADS1292R 芯片输出是 24 位，但数据库的精度不够，因此在 vivado 仿真测试时，只用了 5 位的数据，以此跟 matlab 仿真数据对比。完功能能证明后改为 24 位数据再做仿真。随后是 vivado 仿真需要调用心电数据，而以往的调试只是生成简单的方波，解决办法是在 matlab 中将数据的数据存为 txt 文件，在 testbench 里引用，从而达到了目的。另外 vivado 只能接收二进制或十六进制之类的数据，但心电数据含有负数，不能直接调用 dec2bin 函数。解决方案是在

matlab 里对所有数据加上最小值的绝对值，即把所有数据整体调整到 0 及以上，在 testbench 里再减去这个最小值的绝对值，以此达到了传输数据的目的。

7. 蓝牙模块调试经验总结：设计蓝牙部分时，首先参考了上学期数电实验的蓝牙例程，测试时却遇到了困难，主要是板子响应与教程对不上，后来经过询问杨立昆同学以及查阅源码，发现是教程中对拨码开关和 LED 灯的表述有歧义，和板卡上的丝印并不完全符合，实际上板卡的响应是没问题的。此外我试用了三个手机端的 BLE 调试助手，均无法搜索到板卡对应的设备，请教杨同学后下载了一个新的调试助手，成功连接到板卡并传输数据。此外做蓝牙模块和滤波模块对接时，有一点感悟就是调试好的代码工程最好不要再修改，宁可再多做一个模块。大修改已写好的工程可能带来一些未知的错误，比如我在修改 receiver 时遇到了与它相连的模块出错，而且 FIFO 核也出现了时序错误，调试会花费很多时间，与其大改不如外接一个新模块。

8. PCB 绘制经验总结：首先是型号的选取和成本的考虑。就像前面所说的，型号不同导致封装不同，对 pcb 空间的占据就不同，这是影响排版的一个因素。其次是对立创 pcb 设计软件的掌握，我们上个学期学过 pcb 设计，但并不是立创软件，需要重新熟悉，不过立创比 pcb editor 使用起来更加方便，这方面难题比较少。其次，也是我认为最重要的一点是排版和设计，对电路的美化和优化。这涉及到 pcb 空间资源的分配，涉及到不同层的走线、电路模块的区分等方面，最终影响的是芯片的效果。设计排版，最重要的是合理分配区域，尽量不要让电路元件和导线全部堆积到一个区域，否则结果就是 drc 疯狂报错，导致往往需要在设计完成后再次大改，重新经历排列和设计过程。一些细节但同样重要的部分，例如对芯片管脚的对应。有些导线的连接疯狂交叠，即使是 2 层走线有时也无法承受，这时候可以将导线连在另外一个更近的，相同网络的元件上，最好的例子就是地线网络，有时候飞线能够起到不错的指导作用。最后是采集电极的选择，根据老师的建议选用实心填充的焊盘，重点是实心，否则得到的圆形只是丝印的一个图案。其他的诸如敷铜、DRC 排查等方面，只要熟练掌握了立创软件的使用，都不算困难。

9. SPI 经验教训：我们在上个学期学习过利用 fpga 实现编程的知识，但是 spi 协议不在学习范围之中。不过我们可以通过和 spi 类似的一个 uart 协议，类比地理解 spi 知识。不过和 uart 相比，spi 的通信协议无论是程序的编写还是程序的调测试上都有着巨大的不同。在编写 spi 程序之前，我们需要确定的是 spi 的采样时钟，多久进行一个位数据的采集；确定寄存器的位宽和主从机的选取等基础协议。接下来，我利用 csdn 等网站的相关知识进行学习，利用给出的实例代码进行修改，并在电脑上实现 tb 仿真，检查时序。在仿真成功后，我开始阅读 fpga 的 ego1 手册，得到对 fpga 左方排针对应接口编号的了解，然后

编写 xdc 文件，将接口引出，实现测试和真正的收发信号功能。除了 spi 的测试，还有一项工作是 spi 只是项目对 fpga 功能使用的一部分。除此之外，还有实现 fpga 和手机的蓝牙连接，还有对收到的信号进行滤波处理，都要在 fpga 中进行，这些工作是其他同学实现，但是相对应的接口依然要留下。

不过最大的问题也出现在 spi 的实际测试当中。虽然 tb 仿真很理想，但实际上测试却是困难重重。电脑特性导致，所有的 usb 均是以 uart 的形式传输，spi 四根线的设置也导致无法轻易通过电脑和串口助手来实现对信号的追踪和监测。我们考虑过使用 SX24M8 逻辑仪检查对应接口的信号波形，但是发现逻辑仪最终还是需要利用串口助手，通过发送信号检查对应接口信号。可是 pc 环境下，并没有任何一款串口助手可以支持数据以 spi 协议传输，结果就是，我们既不能确定串口助手发送的信号有无进入 fpga 板中，也无法验证 spi 程序的运行是否正确，因此，我们急需要更好的 spi 测试方法来测试 spi 程序。

## 八、待优化与待改进方案

1. PCB 原理图修改 ADS1292R 芯片的封装，重新投板，降低后期焊接与电路测试的风险。
2. PCB 板与 FPGA 间的 SPI 通信问题的解决
3. 心电图小程序性能的提升，比如蓝牙接收数据的速度、绘图刷新的速度、动态显示的更优实现方案
4. 心率算法的改进，加入异常心电的诊断功能。
5. 对于低频的滤波效果不是很好，可以再耦合一个高通滤波器。
6. 位数转换与数字滤波模块需要细化衔接，数字滤波应考虑实际信号是否采用补码、每一位代表 2 的多少次方，与此同时需要对数字滤波和位数转换的参数做小修改。

## 九、参考文献

- [1]. NATLINEAR, LN2054 datasheet, Feb. 24. 2016.
- [2]. NATLINEAR, LN61C datasheet, Mar. 2. 2016.
- [3]. Texas Instrument, TPS61099x datasheet, Jul. 2016 [Revised May 2018].
- [4]. Texas Instrument, TPS61027DRCR datasheet, Sept. 2003 [Revised Feb. 2016].
- [5]. Advanced Monolithic Systems, AMS1117 datasheet, Sept. 2012.
- [6]. Texas Instrument, ADS1292R datasheet, Dec. 2011 [Revised Sept. 2012].

- [7]. Texas Instrument, "ADS1x9xEKG-FE Demonstration Kit user guide", Dec. 2011 [Revised Apr. 2012].
- [8]. 周雅琪. 便携式心电采集系统及实时分析算法研究[D].浙江大学,2014. 4.
- [9]. 彭保基. 基于蓝牙及 Android 的便携式心电仪的设计与实现[D].吉林大学,2014.
- [10]. TI 模拟前端的生理电信号采集芯片 ADS1292R 的使用 - 子傲代码魔法-小鑫的技术流水账. (2020). Retrieved 9 April 2020, from <https://www.ziaostudio.com/ti%E6%A8%A1%E6%8B%9F%E5%89%8D%E7%AB%AF%E7%9A%84%E7%94%9F%E7%90%86%E7%94%B5%E4%BF%A1%E5%8F%B7%E9%87%87%E9%9B%86%E8%8A%AF%E7%89%87ads1292r%E7%9A%84%E4%BD%BF%E7%94%A8.html>
- [11]. SPI 协议详解\_操作系统\_嵌入式 Linux-CSDN 博客. (2020). Retrieved 25 April 2020, from [https://blog.csdn.net/weiqifa0/article/details/82765892?tdsourcetag=s\\_pctim\\_aiomsg](https://blog.csdn.net/weiqifa0/article/details/82765892?tdsourcetag=s_pctim_aiomsg)
- [12]. SPI 协议的 FPGA 实现 (二) SPI 协议的 FPGA 实现\_碎碎思的博客-CSDN 博客. (2020). Retrieved 27 March 2020, from [https://blog.csdn.net/Pieces\\_thinking/article/details/98474843?ops\\_request\\_misc=%257B%2522request%255Fid%2522%253A%2522158778632919725247645219%2522%2522C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request\\_id=158778632919725247645219&biz\\_id=0&utm\\_source=distribute.pc\\_search\\_result.none-task-blog-2~all~first\\_rank\\_v2~rank\\_v25-11&tdsourcetag=s\\_pctim\\_aiomsg](https://blog.csdn.net/Pieces_thinking/article/details/98474843?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522158778632919725247645219%2522%2522C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=158778632919725247645219&biz_id=0&utm_source=distribute.pc_search_result.none-task-blog-2~all~first_rank_v2~rank_v25-11&tdsourcetag=s_pctim_aiomsg)
- [13]. SPI 通讯逻辑分析仪测试总结\_嵌入式\_yuanmeixiang 的专栏-CSDN 博客. (2020). Retrieved 14 May 2020, from [https://blog.csdn.net/yuanmeixiang/article/details/77689021?tdsourcetag=s\\_pctim\\_aiomsg](https://blog.csdn.net/yuanmeixiang/article/details/77689021?tdsourcetag=s_pctim_aiomsg)
- [14]. 魏华卓. 同步 12 导联心电图综合分析软件的设计与实现[D].清华大学,2014.
- [15]. 牛传莉. 心电信号预处理和波形检测算法的研究[D].北京交通大学,2009.
- [16]. 唐国栋. 基于小波变换心电信号自动分析技术的研究[D].中南大学,2008.
- [17]. Goldberger, A., Amaral, L., Glass, L., Hausdorff, J., Ivanov, P. C., Mark, R., ... & Stanley, H. E. (2000). PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. Circulation [Online]. 101 (23), pp. e215–e220.
- [18]. 实战分享, 教你蓝牙在小程序中的应用. (2020). Retrieved 27 April 2020, from <https://www.cnblogs.com/qcloud1001/p/7717860.html>.

- [19]. 小程序获取连接蓝牙设备所有特征值(getBLEDeviceCharacteristics) - 微信小程序蓝牙教程 - V型知识库. (2020). Retrieved 25 April 2020, from <https://www.vxzsk.com/1868.html>
- [20]. 微信小程序连接蓝牙 ble 教程(目录)\_移动开发\_无聊达的博客-CSDN 博客. (2020). Retrieved 22 April 2020, from [https://blog.csdn.net/qq\\_34234087/article/details/89202665](https://blog.csdn.net/qq_34234087/article/details/89202665)
- [21]. 查找数组中的最大值（最小值），及相应的下标 - 小白呀白菜 - 博客园. (2020). Retrieved 10 May 2020, from <https://www.cnblogs.com/snowbxp/p/12418657.html>